NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

THESIS

DecisionNet:
A DATABASE APPROACH

by

Steven H. Earley

September 1996

Thesis Advisor: Hemant K. Bhargava

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1996	Master's Thesis

4. TITLE AND SUBTITLE
DecisionNet: A DATABASE APPROACH

5. FUNDING NUMBERS

6. AUTHOR(S) Earley, Steven H.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

This thesis describes the database design and implementation for DecisionNet -- a distributed decision support technology server for the World Wide Web. The main premise of DecisionNet is that decision support technologies can be utilized by consumers as services over the World Wide Web instead of being purchased as stand-alone products. In this sense, DecisionNet performs the role of an "agent," facilitating transactions between consumers and providers.

All of DecisionNet's functions involve some form of data lookup and modification, as well as common fields of data for similar classes of entities. As such, a database approach seems appropriate for DecisionNet. With this approach, the interaction of database queries with scripting languages can facilitate remote execution of decision support software.

The DecisionNet prototype developed as a result of this research involves the use of a relational database that is directly accessed via Common Gateway Interface (CGI) scripts. These CGI scripts are invoked by users with a simple web browser. This thesis contains a description of "agent" models for transactions, the relational database design, a description of all CGI scripts, and development of a user interface for the system.

14. SUBJECT TERMS DecisionNet, Decision Support, Relational Database, World Wide Web, Common Gateway Interface

15. NUMBER OF PAGES 258

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298-102

Approved for public release; distribution is unlimited.

DecisionNet:
A DATABASE APPROACH

Steven H. Earley
Lieutenant, United States Navy
B.S.B.A., The Ohio State University, 1988

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
September 1996

Author: _____
Steven H. Earley

Approved by: _____
Hemant K. Bhargava, Thesis Advisor

_____
Suresh Sridhar, Associate Advisor

_____
Reuben T. Harris, Chairman
Department of Systems Management

ABSTRACT

This thesis describes the database design and implementation for DecisionNet -- a distributed decision support technology server for the World Wide Web. The main premise of DecisionNet is that decision support technologies can be utilized by consumers as services over the World Wide Web instead of being purchased as stand-alone products. In this sense, DecisionNet performs the role of an "agent," facilitating transactions between consumers and providers.

All of DecisionNet's functions involve some form of data lookup and modification, as well as common fields of data for similar classes of entities. As such, a database approach seems appropriate for DecisionNet. With this approach, the interaction of database queries with scripting languages can facilitate remote execution of decision support software.

The DecisionNet prototype developed as a result of this research involves the use of a relational database that is directly accessed via Common Gateway Interface (CGI) scripts. These CGI scripts are invoked by users with a simple web browser. This thesis contains a description of "agent" models for transactions, the relational database design, a description of all CGI scripts, and development of a user interface for the system.

TABLE OF CONTENTS

ACKNOWLEDGMENT

The author would like to acknowledge the financial support of the Army Artificial Intelligence Center under Grant MIPR6GNGS00087.

The author wants to thank Professors Bhargava and Sridhar for their expertise and guidance in performing this research. A special thanks to the author's wife, Jennifer. Her love, support, and understanding have been instrumental in making this educational experience a success. This thesis is dedicated to her.

I. INTRODUCTION

This thesis describes the database design and implementation for DecisionNet -- a distributed decision support technology server for the World Wide Web. DecisionNet uses the Hypertext Transfer Protocol (HTTP) and the Common Gateway Interface (CGI) to act as a broker between "providers" (owners of decision support technologies) and "consumers" (users of decision support technologies). These decision support technologies may include data sets, model schemas and instances, solvers, modeling environments, and decision support environments (Bhargava et al., 1995).

A. THE DecisionNet CONCEPT

The main premise of DecisionNet is that consumers should be able to utilize decision support technologies as services over the World Wide Web (WWW), instead of purchasing them as stand-alone products. By providing decision support technologies over the WWW (either for free or for a nominal usage fee), users would be able to tap into the power of such technologies without the normally prohibitive costs involved with purchasing or downloading and installing a specific product. The DecisionNet prototype serves as an interface between consumers and providers of decision support software, so that interactions with multiple technologies may take place at one convenient location. In this sense, DecisionNet performs the role of an agent, facilitating transactions between consumers and providers of decision support technologies.

As an agent, DecisionNet should perform several functions that are common among specific classes of entities. For instance, a consumer will register into DecisionNet, entering such information as his name, e-mail address, and a password. A provider will register into the system in the same manner. DecisionNet must allow for the storage of such information, as well as provide a means for authentication of valid users. Additionally, providers will also want to register their products for use by consumers, so DecisionNet should be able to perform such functions as classification, indexing, and execution of these technologies.

All of these functions involve a change of state in certain entities. For example, a technology being executed would become an *active* technology, identified not only by the technology's name or provider, but also by the particular instance of a consumer using the technology at a given time. During registration, a consumer (without access to DecisionNet) would become a *registered* consumer (with a unique password and access to DecisionNet's consumer-related functions).

B. TYPES OF DATA REQUIRED

In order to accomplish the functionality desired in a system such as DecisionNet, certain data must be acquired and maintained about the entities involved. For instance, consumers' and providers' names and e-mail addresses must be known in case the system administrator would need to contact them. For technologies, a Uniform Resource Locator (URL) address is necessary for access. For an "active" entity (i.e., actively involved in some process within the system), it is important to know the time the entity became active, as well as the latest time of any further activity.

Any data that is required will generally be similar in nature for similar entities. All consumers and providers have names, and all of them should have passwords to enforce system security. All technologies should have a URL address, as well as some descriptive information to help consumers in deciding whether a particular technology is right for them. If this information could be grouped together into one location, users could have access to a powerful repository of decision support technologies at a fraction of the cost of purchasing individual technologies.

C. A DATABASE APPROACH FOR DecisionNet

All of the functions previously mentioned involve some form of data lookup and manipulation (addition, modification, and deletion), as well as common fields of data for similar classes of entities (consumers, providers, and technologies). As such, it makes sense

to store this information in a database, with access to this information made through the use of a database management system (DBMS). Assuming that the actual decision support technologies will reside and execute on the providers' own machines, the processing of data in DecisionNet can be reduced to a series of relatively simple operations (such as SQL queries) on database tables. DecisionNet is specifically designed to be a World Wide Web application; this linkage between a DBMS and the WWW is a relatively unexplored aspect of Web applications.

The current prototype of DecisionNet, developed as a result of this research, is the second major prototype for this distributed decision support system. The original DecisionNet system (Bhargava et al., June 1995; King, 1995) used an HTML-based search engine with scripts written in Perl to accomplish its functionality. The new prototype uses a relational database design, with related data for each entity stored in the form of database tables and data manipulation conducted dynamically through a powerful database engine. This thesis focuses on the DecisionNet database design and database queries, their ability to perform all necessary functions for the system, and the interaction of these queries with scripting languages to facilitate remote execution of decision support software over the World Wide Web.

D. INDEPENDENT AND EXCLUSIVE TECHNOLOGIES

It is important to make a distinction between independent and exclusive technologies as defined for DecisionNet. *Independent* technologies are those for which providers must craft the WWW interface and execution; these technologies can therefore also run independently of DecisionNet. Additionally, providers must program overhead functions such as user registration and accounting, authentication, and billing. *Exclusive* technologies are those for which DecisionNet agents provide the WWW interface, execution control, and all overhead functions; these technologies must therefore be run exclusively through the DecisionNet system. (Bhargava, 1996)

This thesis (and the resulting prototype system) focuses on the independent technology aspects of DecisionNet. The database design and user interface for DecisionNet includes several features that are intended for future expansion once the system is able to accept exclusive technologies. Such features are denoted if they are not currently available. Furthermore, the author assumes that DecisionNet will remain an entirely "free" service for the near future. Financial accounting information is not included in the database design for users, but it is mentioned as a topic for further research.

E. OUTLINE FOR REMAINDER OF THESIS

Chapter II describes the various entities involved in the DecisionNet system, then it delineates the various functions required of a system such as DecisionNet. The chapter continues by discussing DecisionNet as an "agent" process, where the state of an entity (e.g., a consumer) is modified by an agent receiving a particular message. Chapter III contains the database design for the system, including a description of all database tables, the data model, and referential integrity constraints. Chapter IV describes the CGI scripts that are used to perform DecisionNet's functions. Chapter V discusses the development of a user interface, and provides the reader with a "tour" of the prototype system. Finally, Chapter VI provides a summary of this thesis research and describes some of the further research opportunities for DecisionNet.

II. DecisionNet AGENTS AND THEIR MODELS

This chapter describes the DecisionNet system in terms of a series of agent models. These models form the foundation for the DecisionNet database design, with the "agents" actually representing processes that act to modify the database based on specific messages passed between users and the system.

A. DESCRIPTION OF DecisionNet ENTITIES

Before beginning a discussion of the agent models, a brief description of each DecisionNet entity is appropriate. These entities are the objects about which the DecisionNet system must maintain data.

- **Consumer:** a person who is registered for the primary purpose of using DecisionNet technologies.
- **Provider:** a person (or organization) who is registered for the primary purpose of providing technologies for consumers' use.
- **Technology**: a registered decision support technology that is able to operate over the WWW through the DecisionNet interface.
- **System Administrator:** a member of the DecisionNet development team who is responsible for system maintenance and monitoring.
- **Active Consumer:** a consumer who is actively logged into DecisionNet.
- **Active Provider:** a provider who is actively logged into DecisionNet.
- **Used Technology:** a technology (independent or exclusive) that has been used by a consumer.
- **Active Exclusive Technology:** an exclusive technology that is currently being used through DecisionNet .

- **Active Graph Node:** a node of an active exclusive technology, indicating the particular point at which a consumer is working with the technology.

- **Technology Graph:** a listing of an exclusive technology's nodes for data entry and execution; describes the order in which a consumer must input data for the technology to function properly.

B. MAJOR FUNCTIONS FOR DecisionNet

As mentioned earlier, DecisionNet performs several functions that are common among specific classes of entities. These functions, explained below, can be divided into four groups. Consumer-related functions are those functions that are specifically geared toward the users of decision support technologies. Provider-related functions are those functions that are specifically geared toward the owners of decision support technologies. System administrator functions are typical database administrator functions that are not available to the other users. The final group is a "general" category, where the functions apply to all types of users (consumers, providers, and system administrators).

1. Consumer Functions

- **List Technologies:** provide a listing of all registered technologies, with links available for consumers to execute the technologies.

- **Execute Technologies:** provide a means to directly execute a technology using the DecisionNet interface.

- **Technology Search and Retrieval:** allow consumers to search for technologies using specified criteria.

- **Consumer Account Information:** provide a means for consumers to review financial accounting data related to their use of DecisionNet.

2. Provider Functions

- **Register Technologies:** provide an interface to enter information about technologies into the DecisionNet database.

- **Modify Technology Information:** allow providers to review and modify data about their registered technologies.

- **Withdraw Technologies:** allow providers to remove technologies from the DecisionNet database if desired.

- **Browse Taxonomy:** allow providers to view the taxonomy used in indexing technologies.

- **Provider Account Information:** provide a means for providers to review financial accounting data related to their own use of DecisionNet as well as data related to the consumers who are using their technologies.

3. System Administrator Functions

- **View Database Tables:** provide a means to remotely view all tables in the DecisionNet database.

- **Run SQL Commands:** provide a means to remotely execute Structured Query Language (SQL) commands on the DecisionNet database.

- **Remove Users Who Forget to Logout:** allow system administrators to remove registered consumers and providers from an active status. This functionality should also be provided automatically on a periodic basis (i.e., "timeout" capability).

4. General Functions

- **Registration:** allow users to remotely register as consumers or providers. For security reasons, system administrators should only be allowed to register locally.

- **Login, Authentication, and Logout:** provide a means for users to login to DecisionNet, and authenticate a user's identity through a password verification process. Allow users to remove themselves from an active status.

- **Review and Modify Registration Information:** provide a means for users to review and modify their registration data.

- **Withdraw Account:** allow users to remove themselves from the DecisionNet database. For security reasons, system administrators should only be removed locally.

C. FUNCTIONALITY AS AN AGENT PROCESS

All of the functions described above can be represented in the form of agent models that depict an agent's behavior in performing specific operations. Agent functionality is modeled by examining the transactions that take place between agents and other players in the market. Agent behavior during transactions is described by specifying their state space, the messages acceptable in each state, the messages initiated by the agents in particular states, and possible state transitions. (Bhargava et al., 1995)

The discussion of agent models is limited to the overriding scope of this thesis; therefore, models are not developed for exclusive technologies or financial functions. The models will use abbreviated names for the entities described above (and for specific instances of each entity). In this case, the term "entity" is used to describe a database table that stores information on several instances with the same properties. Table 1 summarizes these abbreviations. Furthermore, the symbol $\forall$ is used to indicate the words "for all," and the symbols $\in$ and $\notin$ indicate membership or non-membership of an instance within an entity set, respectively.

A **message** is depicted in the form "(task, reference, data)," where the task is the function to be performed, the reference is the particular element of a set on which to perform the task, and the data is any information that is required to perform the task (Bhargava and Müller, 1995). The reference should be information from a previous related interaction. In the cases where a reference or data is not required, the null set symbol (ϕ) is used instead. Messages are generally functions of an existing state. For example, a user could not possible send a message to modify his consumer account if he is not already registered as a consumer.

Once a message is received and deemed acceptable by an agent, the agent affects some **change in state** for one or more entities. For DecisionNet, a change in state generally means a change to one or more tables in the database.

Entity Set	Abbreviation	Specific Instance of Entity
All Potential Users	U	u
Consumer	C	c
Provider	P	p
Technology	T	t
System Administrator	S	s
Active Consumer	AC	ac
Active Provider	AP	ap
Used Technology	UT	ut

Table 1. Agent Model Entity Abbreviations.

1. Models for Consumer Functions

a. List Technologies

In order to provide a listing of technologies with links for execution, an agent should first verify that a consumer is logged in as a valid user. If the user is valid, the listing may follow. In this case, the rule would be: "For all active consumers in the Active Consumer table, 'list all technologies' is an acceptable message." Using the letter **M** to represent the set of all acceptable messages, the shorthand form of this rule would be: $\forall$ **ac** $\in$ **AC, (listtech, ac,** ϕ**)** $\in$ **M**. No significant change of state would take place.

b. Execute Independent Technologies

The message for executing technologies would be similar to the one for listing technologies in that a consumer must be verified as a valid active consumer. However, a particular technology must be specified by the user before it may be executed. Therefore, $\forall$ **(ac** $\in$ **AC and t** $\in$ **T), (exectech, [ac, t],** ϕ**)** $\in$ **M**. The change of state that occurs as a result

of this message would be to designate the consumer-technology combination as a Used Technology. In shorthand form, the change of state would look like: "Put ac and t in UT," or **UT := UT** $\cup$ **{(ac, t)}**.

c. Technology Search and Retrieval

To allow consumers to search for technologies using specified criteria, a message would still be valid only if the user is an active consumer. The message must also pass the user's specified criteria to the system so that a listing of applicable technologies may be produced. In this case, $\forall$ **ac** $\in$ **AC, (search, ac, criteria)** $\in$ **M**, where "criteria" represents all of the user's choices. This search would obviously be performed on all technologies in the Technology entity set, but again no significant change of state would occur.

2. Models for Provider Functions

a. Register Technologies

For a provider to enter information about technologies, he should first be an active provider. Once that is verified, he may then enter all pertinent data on a new technology. For example, if the provider has two technologies already registered, he may only enter data on his new (third) technology. In this case, $\forall$ **(ap** $\in$ **AP and t** $\in$ **T), (regtech, [ap, t + 1], tech-data)** $\in$ **M**, where "t + 1" represents the provider's newest technology and "tech-data" corresponds to all of the required data for technology registration. Based on this message, a new technology would be registered, or **T := T** $\cup$ **{(t + 1)}**.

b. Modify Technology Information

Modifying a technology's information is similar to registering a technology, except that the provider would modify an existing entity instance rather than create a new one. Also, a provider must have at least one registered technology in order to modify it, and he should be required to enter a password prior to modifying a record. In this case,$\forall$ **(ap** $\in$ **AP and t** $\in$ **T), (modify-tech, [ap, t], [tech-data, pwd(ap)])** $\in$ **M**, where "t" is the particular technology to be modified, "tech-data" is the updated information, and "pwd(ap)" is the active provider's password. Modifying a record would essentially require extraction of data about an entity and rewriting of modified data. This can be modeled as the deletion of a

record ($\mathbf{T := T - \{t\}}$) and the insertion of a new record ($\mathbf{T := T \cup \{t'\}}$). These two operations may be combined to form the state change: $\mathbf{T := T - \{t\} \cup \{t'\}}$, where t' represents the modified technology registration data.

c. Withdraw Technologies

For a provider to remove a technology from the DecisionNet database, the same requirements exist as for modifying technologies. Therefore, $\forall$ **(ap** $\in$ **AP and t** $\in$ **T), (withdraw-tech, [ap, t], pwd(ap))** $\in$ **M**, where "t" is the specific record to remove and "pwd(ap)" is the active provider's password. This message would simply cause the record "t" to be removed from any Technology-related tables. The state change for this case would be: $\mathbf{T := T - \{t\}}$.

d. Browse Taxonomy

Browsing the taxonomy for a provider is virtually identical to listing technologies for a consumer. In order to view the taxonomy, the user must be a valid active provider. An acceptable message could be: $\forall$ **ap** $\in$ **AP, (browse-tax, ap,** ϕ**)**. No significant change of state would take place.

3. Models for General Functions

a. Registration

For registration, it is important to ensure that users are only registered once. Therefore, $\forall$ **u** $\notin$ **C, (register,** ϕ**, u)** $\in$ **M** for a user who wants to register as a consumer, or $\forall$ **u** $\notin$ **P, (register,** ϕ**, u)** $\in$ **M** for a user trying to register as a provider. The state change would be either $\mathbf{C := C \cup \{u\}}$ or $\mathbf{P := P \cup \{u\}}$, again depending upon the type of user.

b. Login, Authentication, and Logout

An agent to allow users to login to DecisionNet must authenticate a user's identity through a password. In this case, $\forall$ **u** $\in$ **C, (login,** ϕ**, [u, pwd(u)])** $\in$ **M**, where [u, pwd(u)] is the user's login code and password. Similarly for a provider login, $\forall$ **u** $\in$ **P, (login,** ϕ**, [u, pwd(u)])** $\in$ **M**. The state change involved would be to make the user an active user: $\mathbf{AC := AC \cup \{u\}}$ for a consumer or $\mathbf{AP := AP \cup \{u\}}$ for a provider. Logging out would simply remove the user from an active status. For a consumer, $\forall$ **ac** $\in$ **AC, (logout, ac,** ϕ**,)**

$\in$ **M**. Similarly for a provider, $\forall$ **ap** $\in$ **AP, (logout, ap,** ϕ**,)** $\in$ **M**. The state change that occurs at logout is either **AC := AC - {ac}** or **AP := AP - {ap}**.

c. Review and Modify Registration Information

Modifying a user's account information is similar to modifying a technology. The only requirement is that the user must be a valid active consumer or provider, which in turn implies that the user is also a registered consumer or provider. The user should also be required to enter a password prior to modifying a record. For consumers, $\forall$ **ac** $\in$ **AC, (modify, ac, [newdata, pwd(ac)])** $\in$ **M**, where "newdata" is the updated information, and "pwd(ac)" is the active consumer's password. Similarly for providers, $\forall$ **ap** $\in$ **AP, (modify, ap, [newdata, pwd(ap)])** $\in$ **M**. Modifying a record would again require extraction of data about an entity and rewriting of modified data, so the state change would be **C := C - {c}** $\cup$ **{c'}** for a consumer or **P := P - {p}** $\cup$ **{p'}** for a provider.

d. Withdraw Account

If a user chooses to remove his account with DecisionNet, the agent should simply remove him as both a registered user and as an active user. This will prevent future unauthorized access to the system. In this case, $\forall$ **ac** $\in$ **AC, (withdraw,** ϕ**, ac)** $\in$ **M** for consumers or $\forall$ **ap** $\in$ **AP, (withdraw,** ϕ**, ac)** $\in$ **M** for providers. The state change would simply be the removal of the user from the appropriate tables. For consumers, the state changes would be **C := C - {c}** and **AC := AC - {ac}**. For providers, the state changes would be **P := P - {p}** and **AP := AP - {ap}**.

4. Summary of Agent Models

Table 2 contains a summary of the acceptable messages and changes in state associated with each of the models developed for consumer and provider functions. These agent models, when used in conjunction with the database design described in Chapter III, form the basis for developing all of the DecisionNet scripts to implement the system's functionality. Generally, each individual script is designed to determine the acceptability of a user's message, then it carries out a specific agent behavior based on that message and the current state of the database.

User Type / Function	Conditions	Acceptable Messages (Task, Reference, Data)	State Change(s)
CONSUMER:			
List Technologies	ac ∈ AC	(listtech, ac, ϕ)	none
Execute Technology	ac ∈ AC, t ∈ T	(exectech, [ac, t], ϕ)	UT := UT ∪ {(ac, t)}
Search and Retrieval	ac ∈ AC	(search, ac, criteria)	none
PROVIDER:			
Register Technologies	ap ∈ AP, t ∈ T	(regtech, [ap, t + 1], tech-data)	T := T ∪ {(t+1)}
Modify Technology	ap ∈ AP, t ∈ T	(modify-tech, [ap, t], [tech-data, pwd(ap)])	T := T - {t} ∪ {t'}
Withdraw Technology	ap ∈ AP, t ∈ T	(withdraw-tech, [ap, t], pwd(ap))	T := T - {t}
Browse Taxonomy	ap ∈ AP	(browse-tax, ap, ϕ)	none
GENERAL:			
Registration	u ∉ C	(register,ϕ, u)	C := C ∪ {u}
	u ∉ P	(register,ϕ, u)	P := P ∪ {u}
Login	u ∈ C	(login, ϕ, [u, pwd(u)])	AC:= AC ∪ {u}
	u ∈ P	(login, ϕ, [u, pwd(u)])	AP := AP ∪ {u}
Logout	ac ∈ AC	(logout, ac, ϕ)	AC:= AC - {ac}
	ap ∈ AP	(logout, ap, ϕ)	AP := AP - {ap}
Modify	ac ∈ AC	(modify, ac, [newdata, pwd(ac)])	C := C - {c} ∪ {c'}
	ap ∈ AP	(modify, ap, [newdata, pwd(ap)])	P := P - {p} ∪ {p'}
Withdraw	ac ∈ AC	(withdraw, ϕ, ac)	C := C - {c}, AC := AC - {ac}
	ap ∈ AP	(withdraw, ϕ, ap)	P := P - {p}, AP := AP - {ap}

Table 2. Summary of Agent Models.

III. DATABASE DESIGN

This chapter describes the data model, the database schema, and referential integrity constraints on the DecisionNet database. It concludes with a description of each category of data used in the database.

A. DATA MODEL

The Entity-Relationship (ER) Diagram for DecisionNet is depicted in Figure 1. All of the entities on this diagram correspond to the entities described in Chapter II. The primary entities for this database design are the **consumer**, **provider**, and **technology** -- a consumer uses a particular technology that is owned by a provider. A consumer or provider may also be an **active consumer** or **active provider** while he or she is using the system. Once a consumer uses a technology, the entities will combine to create a **used technology**. A technology will have exactly one corresponding **technology graph**, and zero to many instances of an **active exclusive technology**, which is used by an active consumer. An active exclusive technology will have exactly one **active graph node** at any time, showing the technology's progress as it is executed. The **system administrator** is also shown in Figure 1. Although the system administrator has no direct relationship with any of the other entities on the ER diagram, this person has the ability to manipulate information about any of the other entities.

B. DATABASE TABLES

The DecisionNet database design was initially designed using Salsa for Windows (1994). Appendix A contains the Semantic Object Diagrams generated by Salsa. Appendix B contains the data dictionary.

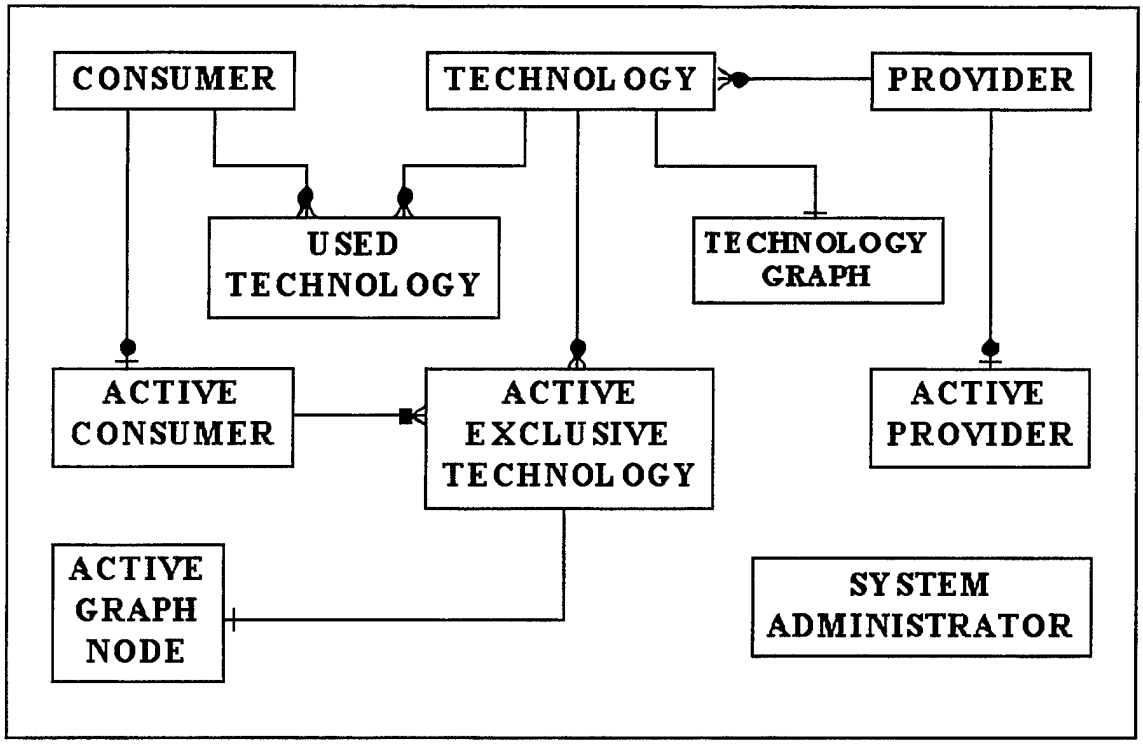


Figure 1. DecisionNet Entity-Relationship Diagram.

The database tables are a logical extension of the Entity-Relationship Diagram from Figure 1. For each entity set in the database, there is a unique table which is assigned the name of the corresponding entity set (Korth and Silberschatz, 1991). In addition to the ten tables formed from the ER diagram, five additional tables provide additional functionality for DecisionNet. The following sections describe the resulting tables, all of which are depicted in Appendix A.

1. Tables Derived From ER Diagram

a. Consumer and Provider Tables

The Consumer table has a unique identifying field named ConsumerID. This is a 15-character identifier, similar to most computer system login names. Additionally, this table contains the consumer's chosen password, last name, first name, and e-mail address. The Provider table is similar to the consumer table, but it only uses one field for the provider's name (individual or business name), and it also includes a field for the Uniform

Resource Locator (URL) address of the provider's home page. The key of the Provider table is ProviderID.

b. Technology Table

The Technology table uses a composite key that includes the ProviderID of the provider who owns a given technology and an additional identifier that gives the "serial number" of the technology. This serial number, named TechID in the table, is assigned based on the number of technologies a provider has registered. There is an additional field for a descriptive (but not necessarily unique) name for the technology. The six classification fields (tObjectType, tProblemArea, tFunctionalArea, tSolutionMethod, tIndType, and tOrgType) are all used to implement the chosen taxonomy for indexing and retrieval (Rogers, 1996). Additionally, there is a field for the URL address of the executable technology, and a field to show whether a technology is exclusive or independent.

c. Active Consumer and Active Provider Tables

These two tables simply contain the respective ConsumerID or ProviderID field, a session starting time, and a last action time. The time fields are obtained by receiving a "time stamp" of the current date and time from the server machine's internal clock. This information is used to monitor a user's session with DecisionNet. In the future, these tables may be used to charge customers on an hourly usage basis.

d. Used Technology Table

This table uses a composite key of the ConsumerID of the consumer using a technology, the ProviderID and TechID of the technology used, and a starting time of usage. There is also a field showing the time the consumer stopped using a particular technology. This information may be used to provide feedback to providers about the consumers who are using their technologies, as well as to provide billing information.

e. System Administrator Table

This table contains only two fields: a login identifier (SysadminID) and a unique password. The System Administrator table is only used for login and authentication before a user is granted access as an administrator.

f. Exclusive Technology Tables

The Active Exclusive Technology table joins an active consumer with a particular exclusive technology. Similar to the Used Technology table, it also contains fields for activation time and the time of last action.

The Technology Graph table lists all of the nodes for an exclusive technology along with each node's respective parent. These nodes help to describe the order in which data must be entered so that a technology will consider the data valid.

The Active Graph Node table keeps track of the particular node for an active exclusive technology at any given point in time. Additionally, this table includes a field for the node's status for data entry (*not ready*, *ready*, or *entered*). The node data field contains the actual data to be passed from the consumer to the technology.

Again, this thesis focuses on the implementation of DecisionNet for independent technologies. The three exclusive technology tables described above are included in this database design for future expansion only.

2. Tables Not Derived from ER Diagram

a. Technology Information Table

The Technology Information table is directly linked to the Technology table, and it contains memo fields (i.e., longer than normal text fields) to describe a technology's purpose and any instructions to the user. This table also contains fields to show the date and time a technology was registered into DecisionNet, as well as a field showing the date and time of the last changes made to a technology's registration data. Although the use of this table provides for some duplicate data (specifically the ProviderID and TechID of each technology), having this infrequently-used information stored on a separate table will provide for more efficient system operation.

b. Consumer, Provider, and Technology Mirror Tables

These tables are identical in design to the Consumer, Provider, and Technology tables, respectively. Each of these "mirror" tables provide a historical listing of information on an entity, even if the entity withdraws from DecisionNet. For example, the

Consumer table only shows those consumers who are currently registered with DecisionNet. The Consumer Mirror table shows all of the registered consumers, plus those who have withdrawn from the system.

c. Taxonomy Table

The Taxonomy table depicts all of the categories developed by Patricia Rogers (1996) for categorizing DecisionNet technologies in a parent-child format. During system development, it became obvious that this taxonomy would be continually evolving as new categories are added. By maintaining the taxonomy on a single table, changes to the taxonomy are more easily implemented and modification anomalies can be avoided. Any script that involves technology registration, modification, or search and retrieval is designed to refer directly to this Taxonomy table. As a result, users are required to use only those categories included in the taxonomy.

C. REFERENTIAL INTEGRITY

A number of referential integrity constraints are designed into the DecisionNet database. The referential integrity rule states that the database must not contain any unmatched foreign key values (Date, 1990). For instance, an active consumer must first be registered as a consumer. Since the Active Consumer table contains the foreign key ConsumerID, there must be a matching value of ConsumerID in the Consumer table. These constraints also help to prevent an unwanted deletion (sometimes known as a "cascading delete"). For example, since a technology first requires the presence of a registered provider, a provider should not be removed from the system until all of his registered technologies are first removed. Table 3 shows a summary of the referential integrity constraints for the database, where the key of the "Parent" table resides as a foreign key in each of the "Children" tables.

Parent	Children
Consumer	Active Consumer, Used Technology
Provider	Active Provider, Technology
Technology	Technology Information, Used Technology, Active Exclusive Technology
Active Consumer	Active Exclusive Technology
Active Exclusive Technology	Active Graph Node

Table 3. Referential Integrity Constraints.

D. CATEGORIES OF DATA

All of the data for DecisionNet may be categorized as either general use data, session data, or archival data. **General use data** includes those items that are necessary for the routine operation of DecisionNet, such as login identifiers and passwords. **Session data** is the information that is required to determine an entity's state at any given point in time. Examples of session data include all of the time stamp fields for active consumers and active providers. **Archival data** includes data that is used to record historical information about each entity. Examples include names, e-mail addresses, and nodes. The data dictionary, shown in Appendix B, includes a column to indicate each data field's category.

IV. CGI SCRIPTS TO PERFORM DecisionNet FUNCTIONS

This chapter describes the Common Gateway Interface (CGI) scripts necessary to facilitate user interaction with the DecisionNet database through the World Wide Web. These scripts are simply platform-specific executable programs that conform to the platform-independent rules of CGI for data exchange. After a brief introduction to the hardware and software chosen for the current version of DecisionNet, this chapter explains the highlights of each script. A complete listing of all of the scripts (written in Object Pascal) is included in Appendix C.

A. PLATFORMS AND PROGRAMMING LANGUAGE

The current implementation of DecisionNet is installed on two machines. The static HTML files (including a basic "shell" in HTML frame format, a "Welcome" page, a "Start DecisionNet" page, an "About DecisionNet" page, a "Contact Information" page, Consumer and Provider Registration pages, and a "Help" file) are all placed on a Sun SparcStation 10 server. This machine serves as the "gateway" to the DecisionNet system. The Paradox for Windows (1994) database and all executable programs are placed on a Pentium-based machine. The programs (and the resulting processor burden) could have been placed on several machines; however, the use of one machine for all programs greatly simplifies implementation and maintenance of the system.

Delphi for Windows (1995) was chosen as the programming language for all DecisionNet CGI scripts. Delphi provides excellent interaction with several database types, and the addition of CGI and CGI Database components (Lynnworth, 1995) allows for a seamless interface between users and the system via the WWW.

B. DESCRIPTION OF CGI SCRIPTS

The CGI scripts are divided into groups, based on the type of user they serve: general user (i.e., not registered or logged in), consumer, provider, or system administrator. These four groups also represent the basic directory structure for DecisionNet on the server -- all general user scripts are located in the "C:\website\cgi-win\dnet\" directory, the consumer scripts are in "C:\website\cgi-win\dnet\consumer\", and so on. Some of the scripts apply to more than one type of user; if this is the case, the script is only described once.

All of the scripts, since they all are designed to operate via CGI, have the same standard structures. Each script has an HTML header section, a body, and a footer section. The **HTML header section** contains error-handling instructions, some of which are specific to the type of server software being used, as well as some statements that identify the program as being CGI-capable. The header sections will not be discussed in the following sections; the reader is invited to review the scripts in Appendix C for the required lines of code. In the **body**, the program receives any required data from the user, then it produces a dynamic HTML page to respond to the user's request. In order to maintain state for a user, the applicable identifier field (ConsumerID, ProviderID, or SysadminID) is passed between scripts using a hidden field. By passing this information in such a manner, any script will be able to verify that it is being called by a valid user. Hidden fields are HTML fields embedded into a form that pass data to CGI scripts without requiring the user to actually enter the data. The **footer section** includes links to related pages within DecisionNet, as well as a statement of authorship.

1. General User Scripts

a. Browse DecisionNet Technologies

The browse script (browse.exe) receives no input data. Once executed, the program returns a listing of all registered DecisionNet technologies, with no links provided to those technologies. In order to receive a listing of technologies with links, the user must register and login as a consumer. The steps are as follows:

- Build a table of technologies using an SQL join of the Technology and Provider tables.

- Send the table to a dynamic HTML page, displaying the technology name, provider name, object type, and problem area for each registered technology.

- Send a footer, including links to register as a consumer or return to the "welcome" page.

b. Null Script

The null script (null.exe) is used for any option on a menu that is currently not available. The present implementation of DecisionNet has three options that are not available: the "keyword search" of technologies on the consumer menu, and the "account information" options on both the consumer and provider menus. These options are all topics for further research in DecisionNet. For these options, the null script does the following:

- Receive either the ConsumerID or ProviderID field from the previous page (Consumer or Provider Menu).

- Send an HTML page that informs the user that his selected option is not available.

- Re-capture the ConsumerID or ProviderID, and send a link to return the user to the appropriate menu.

2. Consumer Scripts

a. Registration Script

The registration script (register.exe) receives its input from the static consumer registration page, and enters a user into DecisionNet as a registered consumer. The password field is verified by requiring the user to type the field twice. The steps are as follows:

- Receive all data fields from the registration page (corresponding to the fields in the Consumer table).

- Verify that the user's selected ConsumerID field does not already exist on the Consumer or Consumer Mirror table. If it does, raise an error and send the user back to the registration page.

- Verify that the password was entered correctly. If it was not, raise an error and send the user back to the registration page.

- If password is correct, enter data into the Consumer, Consumer Mirror, and Active Consumer tables.

- Send an HTML page that informs the user that he is registered as a consumer.

- Capture the ConsumerID and send a link to the Consumer Menu.

b. Login Script

The login script (login.exe) takes its input from the static "Start DecisionNet" page. A registered consumer will enter his ConsumerID and password, and the script will allow him into the system if the password is correct. The login script performs the following:

- Receive the ConsumerID and password fields from user.

- Check Consumer table to ensure the password is correct. If it is not, send an error message to the user, with a link to the Start page to try again.

- Check Active Consumer table to see if user is currently logged in. If he is, send a message stating this fact, and send him to the Consumer Menu.

- If login is valid, send a "Welcome to DecisionNet" message and provide a link to the Consumer Menu.

- Add user's record to the Active Consumer table.

c. Consumer Menu

The consumer menu (menu.exe) is the central point of activity for a consumer. This menu allows the consumer to browse, search, and execute all registered technologies, as well as modify his own registration data. The consumer menu script does the following:

- Receive the ConsumerID field from its calling module. The calling module may be any of the other consumer-related scripts.

- Update the Active Consumer table, replacing the LastActionTime field with the current time stamp.

- **List Technologies:** Allow the consumer to list all registered DecisionNet technologies, sorted by technology name, provider name, object type, or problem area.

- **Access Technologies:** Allow the consumer to directly access a technology by ProviderID and TechID, via the "About" script. The ProviderID and TechID are chosen from drop-down lists that are dynamically generated from the Technology table.

- **Indexed Search:** Provide the ability to search for technologies that match the consumer's choices for each of the six major categories in the taxonomy (object type, problem area, functional area, solution method, industry type, organization type).

- **Keyword Search:** Allow the consumer to search for a technology using keywords (not available).

- **Modify Information:** Provide the ability to modify a consumer's registration information in the database.

- **Account Information:** Provide the ability for a consumer to review his financial account with DecisionNet (not available).

- **Withdraw:** Allow a consumer to permanently withdraw his account with DecisionNet.

- **Logout:** End an active consumer session.

d. List Technologies

The list technologies script (listtech.exe) is similar to the browse script described in the previous section. The only difference is that the List Technologies script provides links for the consumer to execute technologies if desired. This program performs the following:

- Receive the ConsumerID field and the consumer's choice of the field for sorting the result table.

- Update the Active Consumer table.
- Build a table of technologies using an SQL join of the Technology and Provider tables.
- Send the table to a dynamic HTML page, displaying the technology name, provider name, object type, and problem area for each registered technology. The technology name is linked to the "About" script, which is the next step in technology execution.
- Send a footer, including a link to the Consumer Menu.

e. About Technologies Script

This script (about.exe) is the first step in technology execution. When called, this program provides the consumer with information about his chosen technology. The majority of the information provided is from the Technology Information table, including the "purpose" and "comments" memo fields, the date and time a technology was registered with DecisionNet, and the date and time a technology was last updated. Once the consumer has verified that the technology chosen is the correct one, a link is provided to launch the technology. The about technologies script performs the following:

- Receive ConsumerID from the calling module, as well as the ProviderID and TechID of the desired technology
- Update the Active Consumer table.
- Build a table of information for the technology, using an SQL join of the Technology, Technology Information, and Provider tables.
- Send the table on a dynamic HTML page to the consumer.
- Send a footer, with a link to the Execute Independent Technology script (for independent technologies) or the "Null" script (for exclusive technologies, since this option is not available).
- Send a link for the user to cancel and return to the Consumer Menu if desired.

f. Execute Independent Technologies

Once a consumer has reviewed the information provided in the about technologies script and he has decided to execute an independent technology, the link provided will call this module (execind.exe) to record the event. It is important to note that if a consumer leaves the DecisionNet shell then tries to return by reloading a DecisionNet page, his state information will be lost. This situation could easily occur when a consumer executes an independent technology, and he would be forced to log back in to the system to resume access to any consumer-related functions. To help avoid this problem, the execute independent technologies script opens the technology's page in a new browser window, leaving the DecisionNet shell untouched. The consumer is instructed to close the new window when he is finished with the independent technology, then his DecisionNet session may continue as normal. This script proceeds as follows:

- Receive ConsumerID, ProviderID, and TechID as hidden fields from the about technologies script.
- Update the Active Consumer table.
- Add the consumer and technology combination to the Used Technology table.
- Send the consumer to the independent technology in a new browser window.

g. Indexed Search

This script (indexed.exe) receives the consumer's choices of the six major indexing categories from the DecisionNet taxonomy and finds any technologies that match the choices. This is not a "true" indexed search in that it does not find any technologies that are close to (but not exactly) the user's choice; this script only returns exact matches. The user has the option to choose "ALL" for any category, which implies that he has no limiting criteria for that category. Additionally, certain technologies may be registered as "ALL" for a particular category. For example, a technology that is categorized as "ALL" in the tOrgType field implies that the technology is valid for all organization types. The multi-dimensional queries involved with these "ALL" categories can become so complex that the

database engine will return a "capability not supported" error. To alleviate this problem, an approach was taken to perform two smaller queries and combine their results together. The final result is a table of the technologies that match the consumer's request (including technologies categorized as "ALL"), with a link for each pointing to the About Technologies script. The table will only display those fields for which a consumer did not select "ALL." The Indexed Search script performs the following:

- Receive ConsumerID and the six criteria from the Consumer Menu.
- Update the Active Consumer table.
- Perform a query on the Technology table to select any technologies that exactly match the user's request.
- Perform a query on the Technology table to select any technologies that are categorized as "ALL" for any of the user's requested categories.
- Append the result of the second query onto the result from the first query.
- Display the final result as a table, with links for technologies pointing to the About Technologies script.
- Display a link for the user to cancel and return to the Consumer Menu.

h. Modify Consumer Information

These scripts (modify.exe, modifya.exe) work in conjunction to modify a user's information on the Consumer table. The modify.exe program displays an HTML form to collect the user's inputs, and the modifya.exe program takes these inputs and modifies the appropriate tables. The steps involved include:

- Receive ConsumerID from the Consumer Menu.
- Update the Active Consumer table.
- Display an HTML page with a form for the user's information, displaying current values (except the password).
- Send all updated data to modifya.exe.

- If the password entered is invalid, display an error message and send the user back to the Consumer Menu.
- If the password is correct, modify the Consumer and Consumer Mirror tables with the new information.
- Send a link for the user to return to the Consumer Menu.

i. Withdraw from DecisionNet

The withdraw script (withdraw.exe) permanently removes a consumer from the Consumer table, but leaves him in the Consumer Mirror table for historical purposes. The consumer is advised that he must choose a new ConsumerID if he registers again in the future. The steps performed by this script are:

- Receive ConsumerID from the consumer menu.
- Remove consumer's record from the Consumer and Active Consumer tables.
- Send a dynamic page with contact information for comments to DecisionNet's creators.

j. Logout Script

The logout script (logout.exe) removes a user from the Active Consumer table and ends an active consumer session. The steps performed by this script are:

- Receive ConsumerID from the consumer menu.
- Remove consumer's record from the Active Consumer table.
- Send a dynamic page with contact information for comments to DecisionNet's creators.

3. Provider Scripts

a. Provider Scripts that Parallel Consumer Scripts

All of the consumer-related scripts described above, with the exception of the consumer menu and indexed search scripts, have nearly identical counterparts with the same file names in the Provider area of DecisionNet. The only difference in these scripts is that they each pass ProviderID as a hidden field instead of ConsumerID and any link to the Consumer Menu would instead point to the Provider Menu. The code for all of the provider scripts is still included in Appendix C; however, these scripts will not be described further here. The remaining scripts in this section are those that are specifically designed for providers.

b. Provider Menu

Similar to the consumer menu, the provider menu (menu.exe) is the central point of activity for a registered provider. This menu provides the provider with technology-related options (register, update, withdraw, and technology information), account-related options (modify, withdraw, account information, and logout), and two additional options (list technologies and browse taxonomy). The provider menu script does the following:

- Receive the ProviderID field from its calling module. The calling module may be any of the other provider-related scripts.
- Update the Active Provider table, replacing the LastActionTime field with the current time stamp.
- **Register Technology:** Provide the ability to register new technologies.
- **Update Technology:** Allow providers to modify registration data on their technologies.
- **Technology Information:** Allow providers to view information on the consumers who are using their technologies.
- **Withdraw Technology:** Allow providers to remove their technologies from DecisionNet.

- **Modify Information:** Provide the ability to modify a provider's registration information in the database.

- **Account Information:** Provide the ability for a provider to review his financial account with DecisionNet (not available).

- **Withdraw:** Allow a provider to permanently withdraw his account with DecisionNet.

- **Logout:** End an active provider session.

- **List Technologies:** Allow the provider to list all registered DecisionNet technologies, sorted by technology name, provider name, object type, or problem area.

- **Browse Taxonomy:** Allow the provider to view the DecisionNet taxonomy table.

c. Register Technology

The register technology scripts (regtech.exe, regteca.exe) work together to allow a provider to register a new technology into DecisionNet. The regtech.exe program displays an input form to collect registration data. The regteca.exe program accepts the data from regtech.exe, and adds it to the necessary tables in the database. These scripts proceed as follows:

- Receive ProviderID from the provider menu.

- Update the Active Provider table.

- Open the Technology Mirror table to find the last TechID number registered for the provider. If no technologies are registered, assign the number "1".

- Display the HTML form for technology registration data input, displaying the correct ProviderID and TechID fields.

- Display drop-down lists for the six major taxonomy categories, generated dynamically from the Taxonomy table.

- Link the form to regteca.exe and send the data to this script.

- Update the Active Provider table again.
- Append the new record to the Technology, Technology Mirror, and Technology Information tables.
- If the technology is exclusive, send the user to the Register Exclusive Technology script (not available) for further data input.
- If the technology is independent, return a message stating that the technology is registered and send the user back to the provider menu.

d. Update Technology

These scripts (updttech.exe, updtteca.exe, updttecb.exe) enable the provider to modify information pertaining to a registered technology. The updttech.exe script displays a small form that prompts the provider for his password and his choice of technology to update. Only that provider's registered technologies are displayed. The updtteca.exe script displays a form (similar to the technology registration form) showing all of a technology's current data. The provider is instructed to make any desired changes on this form. The updttecb.exe script receives this new data and modifies the appropriate tables. These programs perform the following:

- Receive the ProviderID from the provider menu.
- Update the Active Provider table.
- Prompt the user to enter his password and select a technology to modify.
- If the password is incorrect, notify the user and send him back to the provider menu.
- If the password is correct, call updtteca.exe to display the modification form.
- When the provider changes any of the fields, send the new data to updttecb.exe.
- Update the Active Provider, Technology, Technology Mirror, and Technology Information tables.

- If the technology is exclusive, send the user to the Update Exclusive Technology script (not available) for further data input.
- If the technology is independent, return a message stating that the technology has been updated and send the user back to the provider menu.
- Provide the ability at all stages to cancel without making changes.

e. Technology Information

This script (techinfo.exe) allows the provider to view information on the users who are using his technologies. The program performs the following functions:

- Receive the ProviderID from the provider menu.
- Update the Active Provider table.
- Perform a query on the Used Technology table, selecting those records that show technologies owned by the provider.
- Display the result in a table.
- Return the user to the provider menu

f. Withdraw Technology

These scripts (wdtech.exe, wdtecha.exe) work to permanently remove a technology from DecisionNet. The wdtech.exe program displays a small form that prompts the provider for his password and his choice of technology to withdraw. Only that provider's registered technologies are displayed. The wdtecha.exe program removes the technology from the Technology table, leaving the record in the Technology Mirror table for historical purposes. The scripts proceed as follows:

- Receive the ProviderID from the provider menu.
- Update the Active Provider table.
- Prompt the user to enter his password and select a technology to withdraw.

- If the password is incorrect, notify the user and send him back to the provider menu.
- If the password is correct, call wdtecha.exe to remove the technology.
- Remove the technology from the Technology Table.
- Return the user to the provider menu.

g. Browse Taxonomy

This script (browstax.exe) allows the provider to view the Taxonomy table, sorted by parent for easier viewing. The Browse Taxonomy program does the following:

- Receive the ProviderID from the provider menu.
- Update the Active Provider table.
- Display the Taxonomy table using a query, ordering the table by the Parent field.
- Return the user to the provider menu.

4. System Administrator Scripts

a. Login Script

The system administrator login (login.exe) is similar to the consumer and provider login procedure, except the login form is on a separate static HTML page. Once the user is logged in, a link is provided that points to the system administrator menu.

b. System Administrator Menu

The system administrator menu (menu.exe) is the launching point for the remaining scripts. This menu allows the user to change his system administrator password, view any table in the DecisionNet database, run a valid SQL statement on any table, and manually execute the DecisionNet "timeout" script.

c. Change Password

This script (modify.exe) is similar to the scripts used to modify consumer or provider information. The difference is that the system administrator table has only one field available to modify -- the password. The form for modifying the password is located on the system administrator menu, and it requires the user to type his old password once and his new password twice for verification. The script proceeds as follows:

- Receive old and new password fields from the system administrator menu.
- If any of the passwords are invalid, display an error message and send the user back to the menu.
- If the passwords are correct, display a message stating that the password has been changed.
- Return the user to the system administrator menu.

d. View DecisionNet Tables

This script (viewtabl.exe) enables the user to view any of the tables in the DecisionNet database. The tables are listed on a drop-down list on the menu. The program performs the following:

- Receive the SysadminID and the user's choice of table from the system administrator menu.
- Display the selected table on a dynamic HTML page.
- Display the drop-down list again to give the user the option of selecting another table.
- Provide a link to return to the system administrator menu.

e. Run SQL Statement

This script (runsql.exe) allows the user to type an SQL statement and execute it on any table in the DecisionNet database. A text box is provided on the menu. The script proceeds as follows:

- Receive the SysadminID and the user's SQL command from the system administrator menu.
- For "SELECT" statements, display the resulting table on a dynamic HTML page.
- For any other statement that does not return a result set, display a message stating that the command has been executed.
- If the command is invalid, allow the database engine to display an error message.
- Display the text box again to allow the user to enter another SQL statement.
- Provide a link to return to the system administrator menu.

f. Timeout Script

This script (timeout.exe) is a CGI version of the automated "timeout" program that purges records from the Active Consumer and Active Provider tables for users who have remained inactive for a long period of time. The majority of users who fall into this category are those who leave DecisionNet and forget to logout. The automated script (a non-CGI Delphi program) is scheduled to run hourly on the DecisionNet server. The "manual" version performs the following functions:

- Receive the SysadminID from the system administrator menu.
- Remove any records from the Active Consumer and Active Provider tables where the LastActionTime field is greater than six hours old.
- Display a message to the user showing the number of records deleted from each table.
- Return the user to the system administrator menu.

V. DEVELOPMENT OF A USER INTERFACE

This chapter describes the DecisionNet user interface. To use the system, all a consumer needs is access to the World Wide Web and a frames-capable browser. Even if the consumer does not have a frames-capable browser, he may still use DecisionNet, but he will not be able to access the DecisionNet shell for commonly-used functions. The discussion of the user interface begins by describing the state transition diagram for the system. The chapter concludes with a guided tour of DecisionNet for a typical consumer session. Appendices D and E contain printouts of the pertinent user interface screens for provider and system administrator sessions, respectively.

A. STATE TRANSITION DIAGRAM

The state transition diagram for each type of DecisionNet user -- consumer, provider, and system administrator -- is depicted in Figure 2. This diagram shows the flow of the user interface between the various static HTML pages and the dynamic pages generated by CGI scripts. Static HTML pages are denoted in Figure 2 by an asterisk.

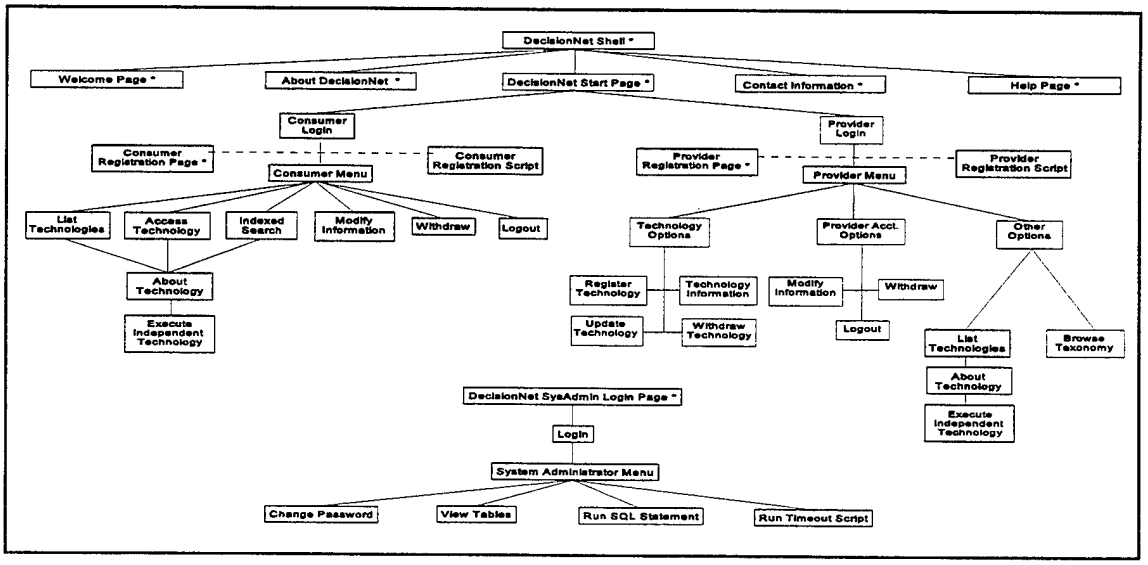


Figure 2. DecisionNet State Transition Diagram.

B. A TOUR OF DecisionNet

The typical consumer enters the DecisionNet system through the shell, which is shown in Figure 3. The DecisionNet shell is a frames-based interface that allows the user access the top-level (not user specific) functions and informational pages for the system. The right-hand frame within this shell contains the DecisionNet "welcome" page. The shell also contains links to the other DecisionNet prototypes being developed at Carnegie-Mellon University and Humboldt University in Berlin, Germany.

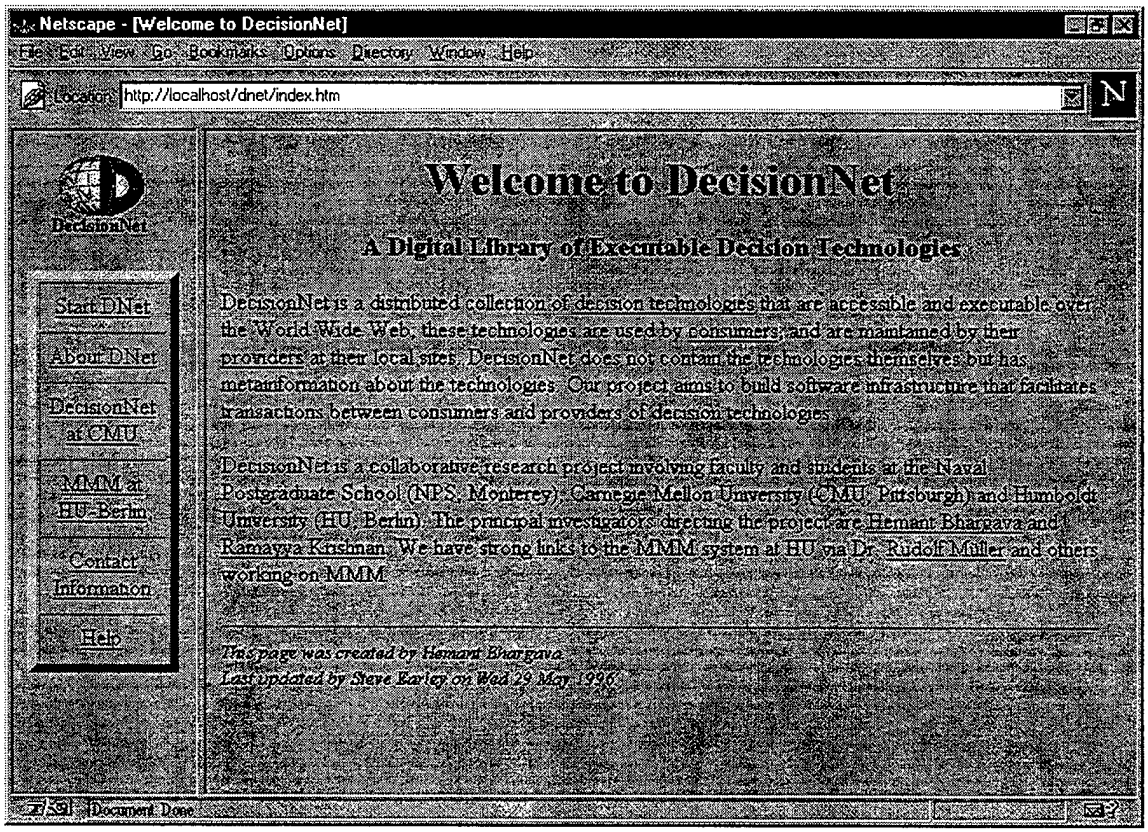


Figure 3. DecisionNet Shell.

The About DecisionNet page, depicted in Figure 4, describes some of the background information and research surrounding DecisionNet. This page also defines the major entities in the system -- consumer, provider, and technology. Links are provided to several other

relevant sites on the World Wide Web. Since this page is fairly long, Figure 4 only shows the top portion. The reader is invited to visit DecisionNet on-line to examine these pages more thoroughly.

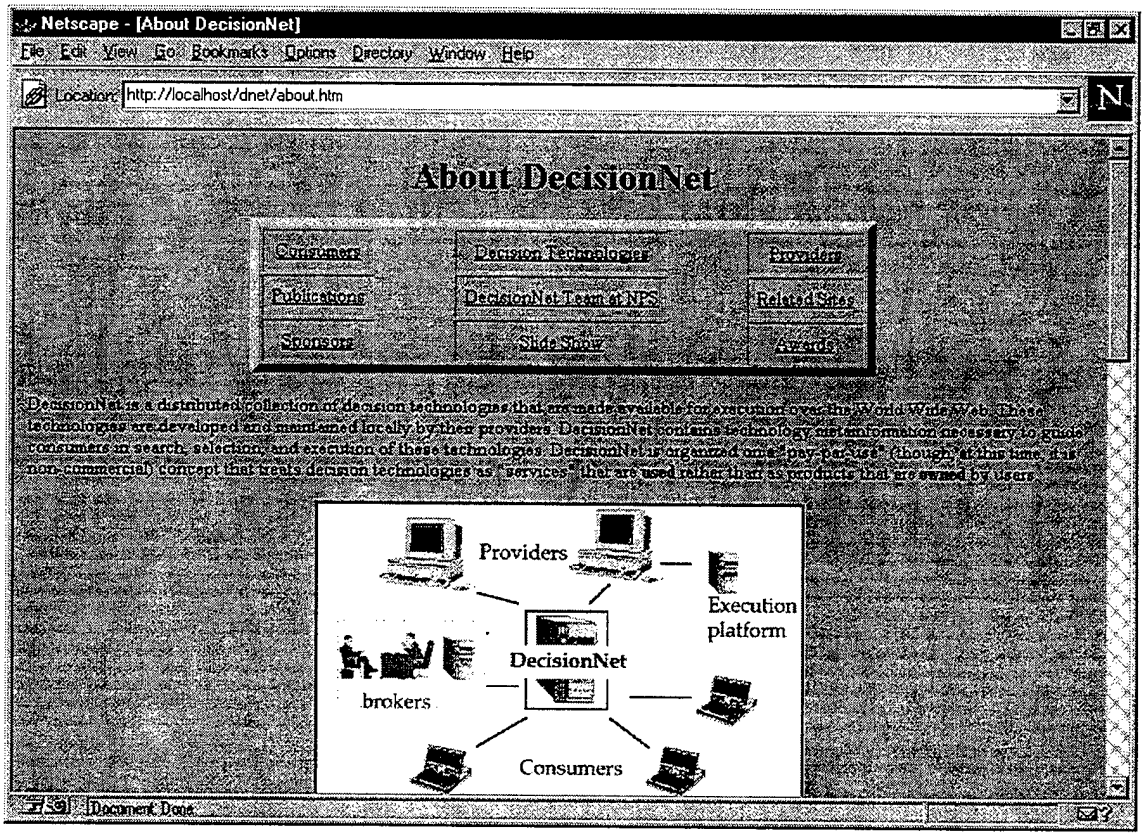


Figure 4. About DecisionNet Page (Top Portion).

The contact information page (not shown) includes pertinent addresses and telephone numbers, including an e-mail address for comments or suggestions. The help page (also not shown) lists several useful tips for the new user.

The Start DecisionNet page, shown in Figure 5, contains the forms for registered consumers and providers to login and begin new sessions. This page also provides links for new users to register as consumers or providers. The "Browse" option allows an unregistered user to view a listing of DecisionNet technologies without being able to execute them.

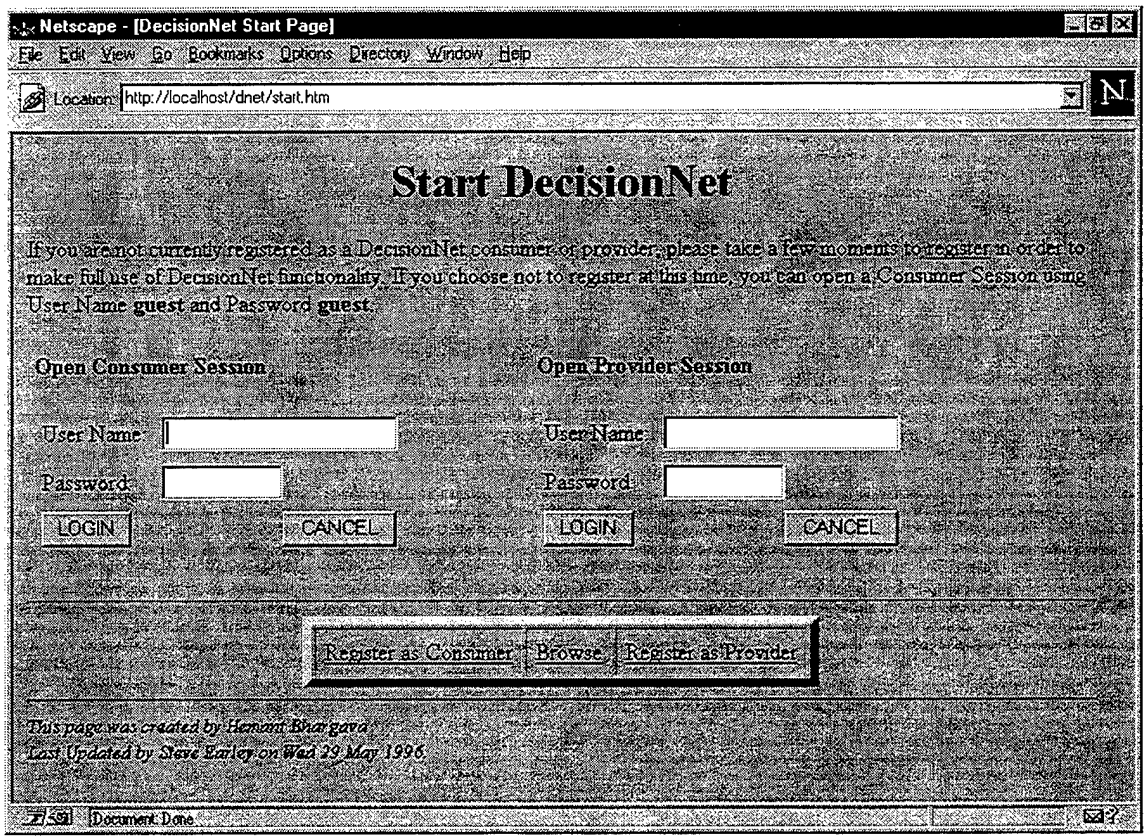


Figure 5. Start DecisionNet Page.

When the consumer follows the "Register as Consumer" link from the start page, he is taken to the static consumer registration page. This page is depicted in Figure 6. The fields in this registration page parallel the structures of both the Consumer and Consumer Mirror tables, except that the consumer is prompted to type his password twice for verification.

Once the consumer has entered all of the required fields, he presses the "Sign Me Up" button. This button calls the consumer registration script, which inputs the user's data into the Consumer, Consumer Mirror, and Active Consumer tables. The script returns a dynamic login confirmation page as shown in Figure 7.

This login confirmation page then directs the user to the consumer menu. The button on this confirmation page represents an underlying form that captures the user's ConsumerID field as a hidden field and sends it as a variable to the consumer menu.

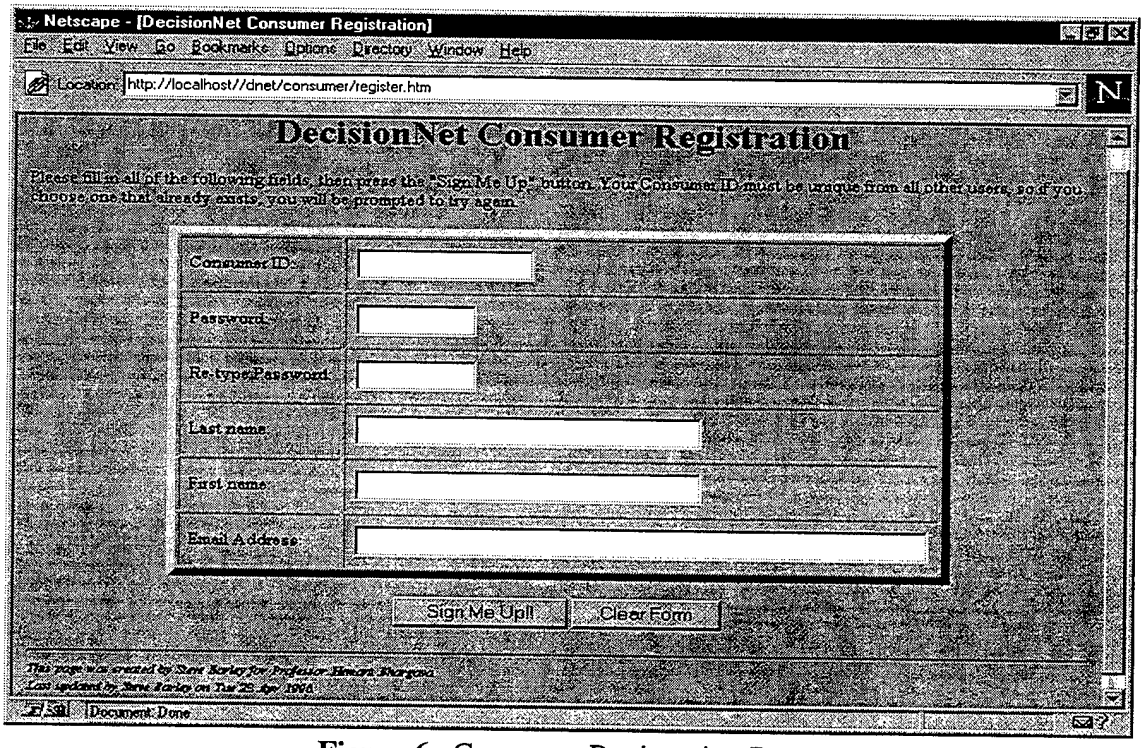


Figure 6. Consumer Registration Page.

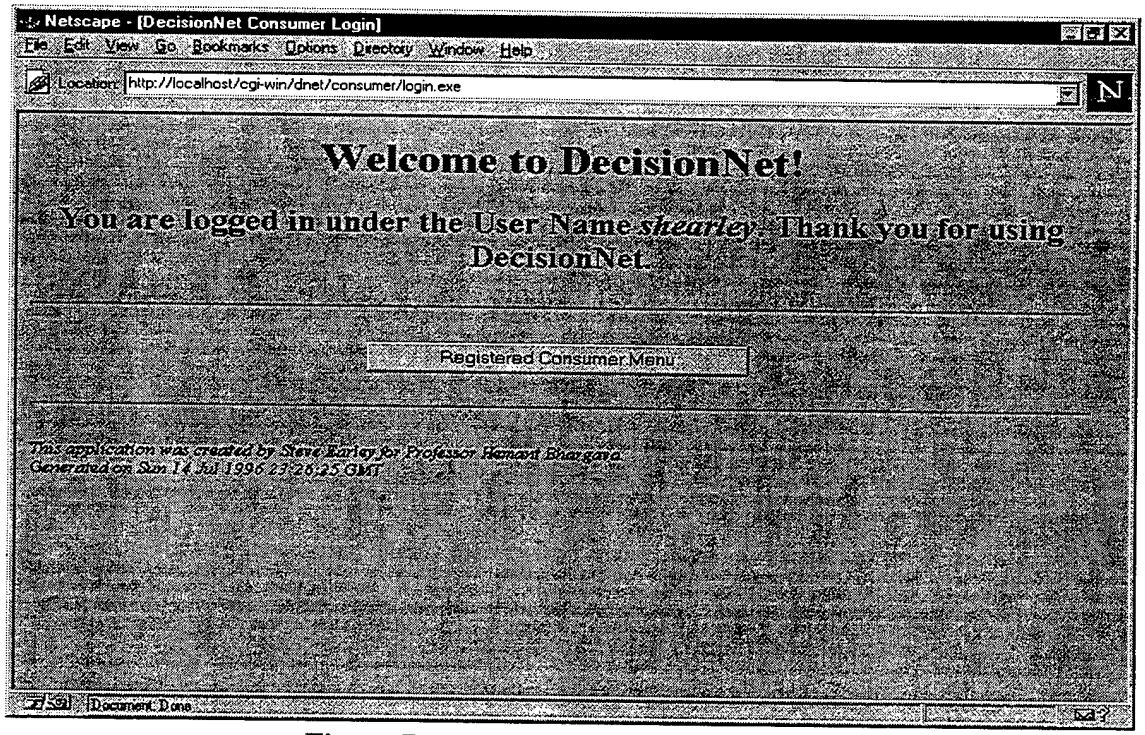


Figure 7. Consumer Login Confirmation.

The consumer menu, shown in Figure 8, contains all of the options available to the registered consumer. Each option on this menu is an individual HTML form that passes the ConsumerID field (along with any other pertinent data) to the option's respective script.

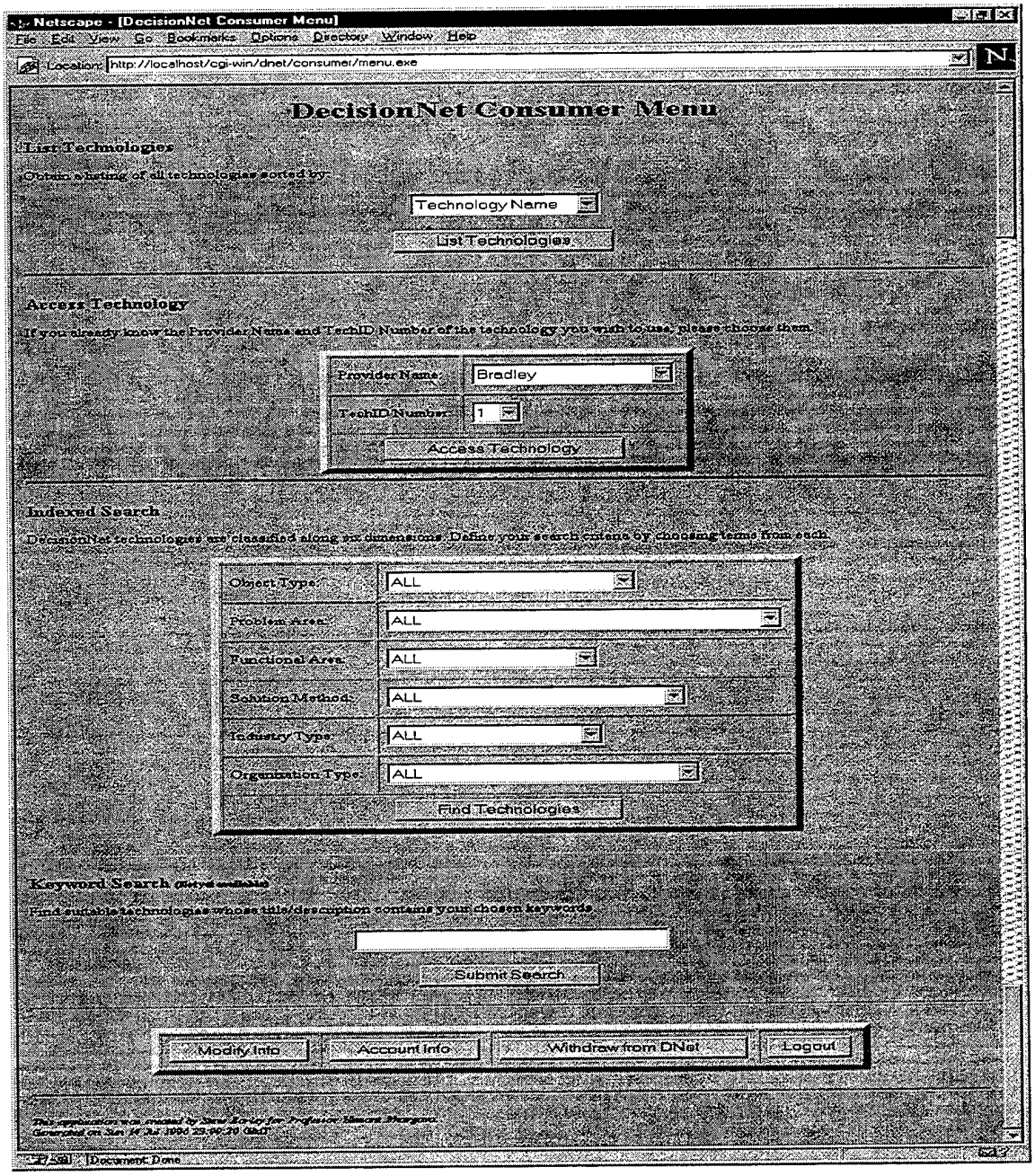


Figure 8. Consumer Menu.

The list technologies script (output not shown) receives the consumer's choice of a sorting key, and returns a table listing all of the registered DecisionNet technologies. The listing includes an individual HTML form for each technology that captures the user's ConsumerID field and sends the user to the "About" script for the technology chosen.

The access technology script (output not shown) simply takes the user's input for a ProviderID and TechID and sends the user to the "About" script for that particular technology. The options for the ProviderID fields and the TechID fields are generated so that all possible ProviderID codes *and* all possible TechID numbers are shown. Therefore, it is possible that the user could select an invalid combination of the two fields (e.g., if a provider only has one registered technology and the user selects that provider with a TechID of "2"). In this case, the script raises an error message informing the user of the invalid combination.

The indexed search script takes the user's choices for each of the six major categories for indexing DecisionNet technologies, and returns a table of technologies that match the user's choices. Each technology in the output table is linked to a form that directs the user to the "About" script. Figure 9 shows a sample output from this script, where the user requested a listing of all technologies where:

- Problem Area is "Asset Selection"
- Functional Area is "Shopping"
- Industry Type is "Consumer"
- Organization Type is "Personal/Individual."

This table that is returned to the user (Figure 9) shows that there is only one technology that exactly matches his criteria. There are, however, three other technologies that may apply to all problem areas, functional areas, industry types, and organization types.

The "About" script receives the ConsumerID field, along with the user's choice of ProviderID and TechID, and returns a page of information about the chosen technology. An example of the script's output is found in Figure 10. Once the user has decided to execute

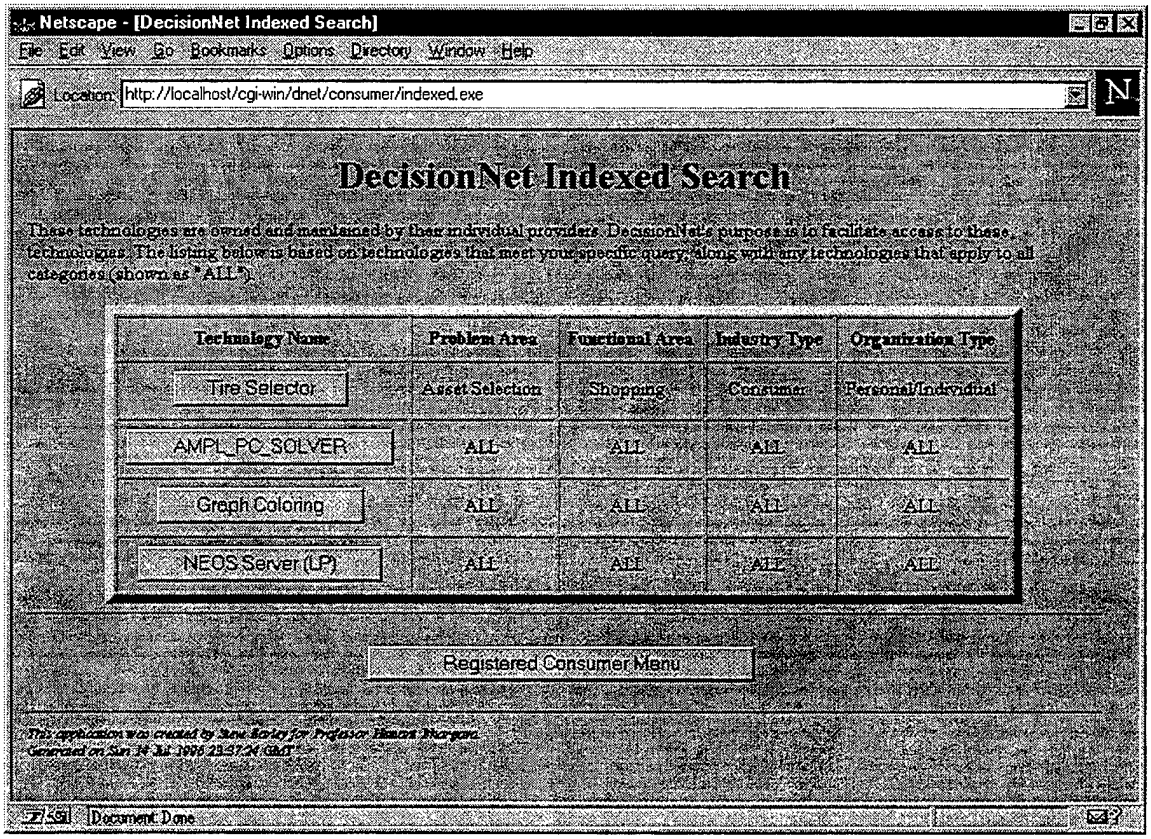


Figure 9. Indexed Search Output Page.

the technology, he selects the link on this output page. The execute independent technology script then opens a new browser window and allows the user to run the technology.

Since the neither the keyword search nor the account information options are available, the forms associated with these options point to the "null" script. An example of the null script's output is shown in Figure 11.

The modify consumer information page (not shown) is very similar to the consumer registration page, except that it is generated dynamically from the modification script. Once displayed, this page shows all of the current values for the user in the Consumer table (except for the password). Once the consumer verifies his changes, a confirmation page is sent and the user is directed back to the consumer menu.

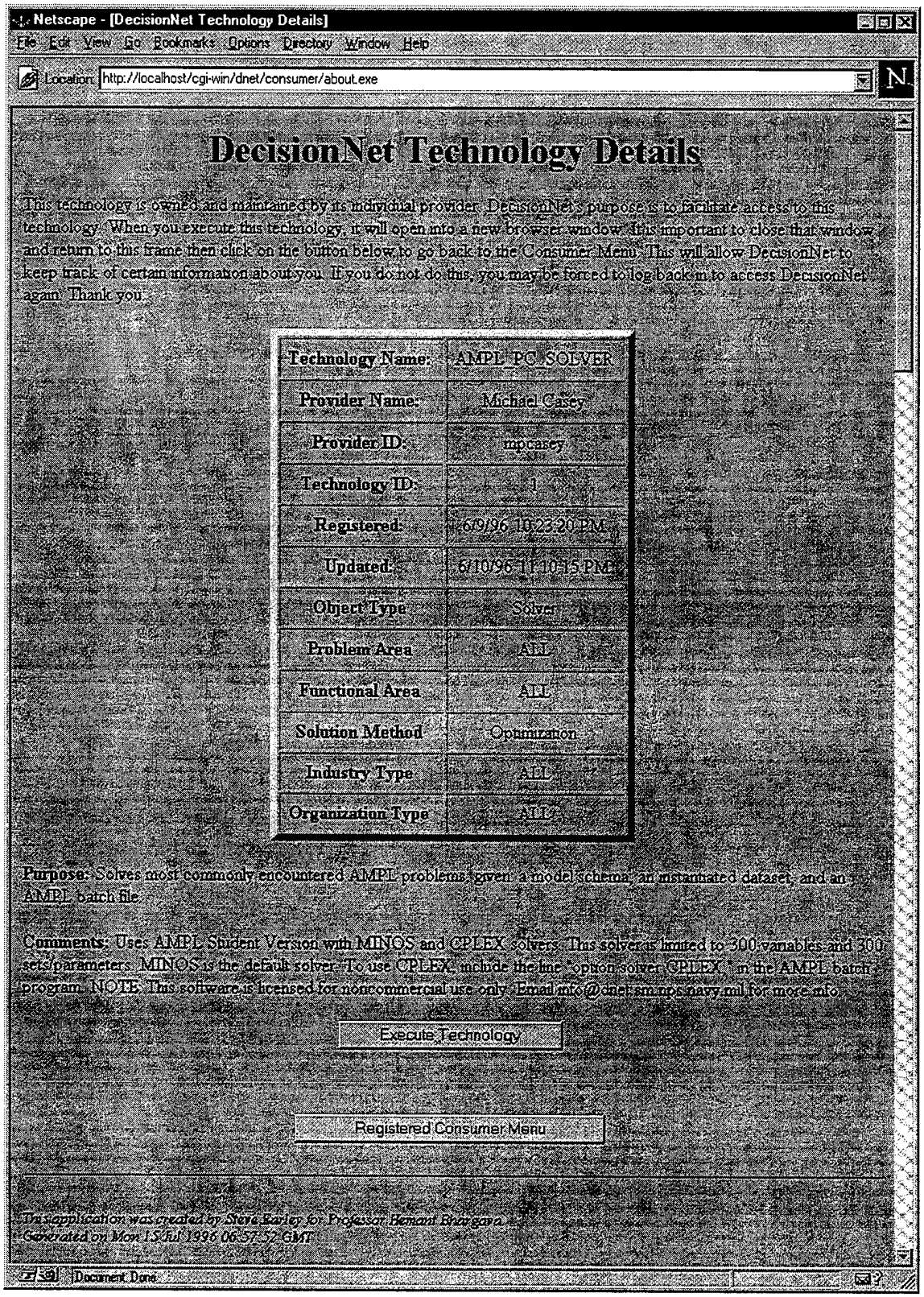


Figure 10. About Technology Script Output Page.

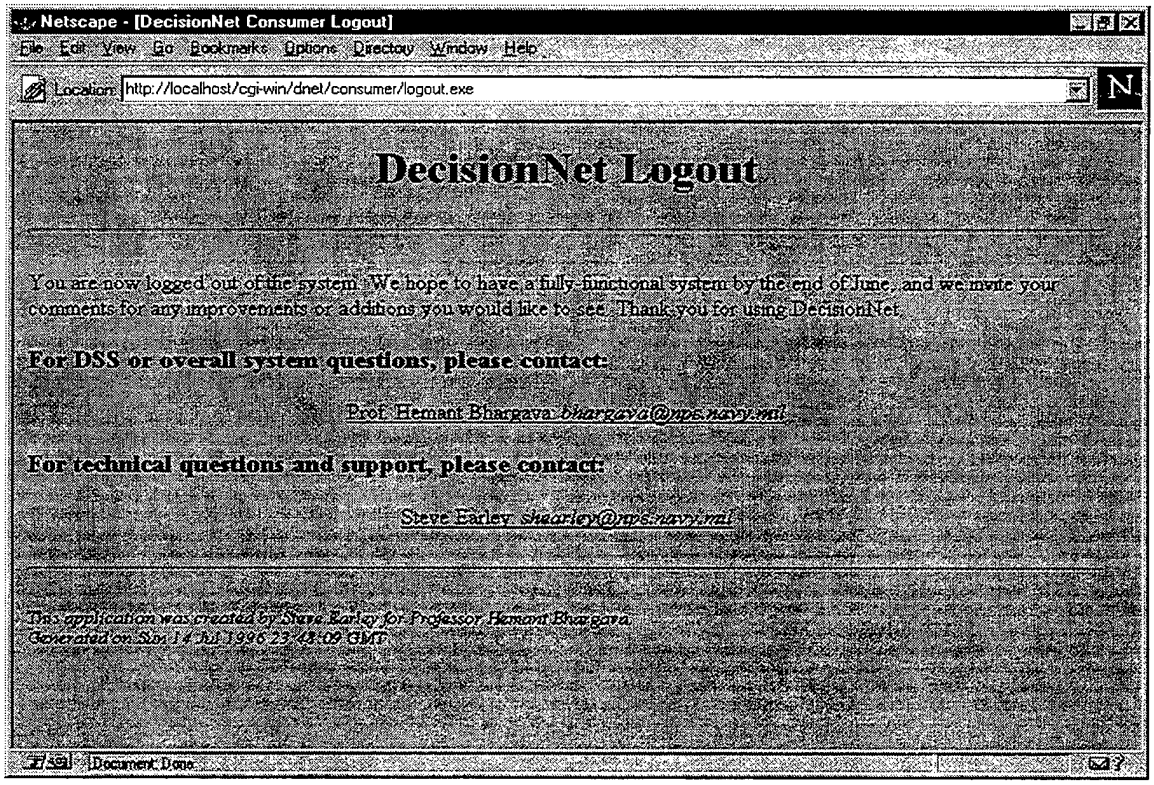


Figure 11. Consumer Logout Script Output Page.

The consumer withdraw script and the consumer logout script are similar in function, and their output pages are also nearly identical. An example of the logout script's output page is depicted in Figure 12.

Printouts of the static and dynamic pages for providers and system administrators can be found in Appendices D and E, respectively. In general, a page is shown in these Appendices only if it is significant and it does not closely resemble any of the consumer pages depicted in this chapter.

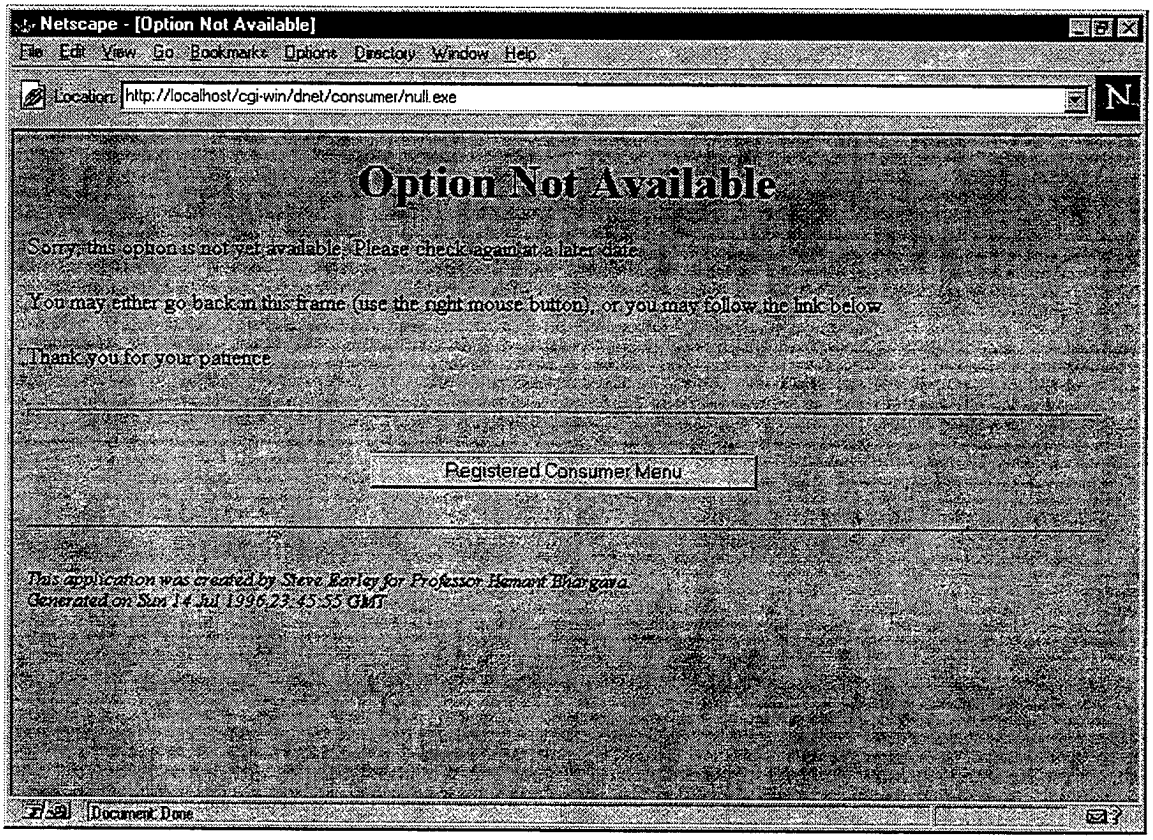


Figure 12. Consumer Null Script Output Page.

VI. CONCLUSIONS

A. LOOKING BACK

This thesis has described the development and implementation of a database prototype for a distributed decision support technology server for the World Wide Web. By providing decision support technologies over a global medium such as the WWW, users are able to tap into the power of decision support technologies without the normally prohibitive costs involved with purchasing or downloading and installing a specific product. The DecisionNet prototype serves as an interface between consumers and providers of decision support software, so that interactions with multiple technologies may take place at one convenient location.

All of the required functions for DecisionNet involve some form of data lookup and manipulation, as well as common fields of data for similar classes of entities. A database approach is a logical solution for organizing and manipulating information for DecisionNet. Since the actual decision support technologies are stored reside on the providers' own machines, data processing for DecisionNet consists of a series of relatively simple operations on database tables.

DecisionNet's functionality as a database system can be described in terms of a series of agent models. These models formed the foundation for the DecisionNet database design and the underlying scripts. The "agents," in essence, represent processes (i.e., scripts) that act to modify the database based on specific messages passed between users and the system.

The CGI scripts developed for this thesis are the product of several hundred hours of programming and compiling, and as such they represent the majority of the research effort. Through the use of rapid application development tools and a few add-on components, programming for CGI applications differs very little from normal object-oriented or structured programming.

By designing the user interface to be fully accessible through virtually any commercial web browser, nearly every World Wide Web user becomes a potential consumer

for the DecisionNet system. This fact can be a strong selling point for technology providers, who are likely to view DecisionNet as a marketing tool for their products. More importantly for consumers, this type of interface further reduces the costs (both in terms of money and time) associated with using decision support technologies.

B. LOOKING AHEAD

1. Future DecisionNet Research

The following are some of the topics surrounding DecisionNet that require further research:

a. Exclusive Technologies

DecisionNet must be expanded to allow for the execution of exclusive technologies. Although this thesis did not focus on the registration and indexing of exclusive technologies, the appropriate database tables have been implemented to make this transition easier. Essentially, research in this area will focus on the automatic generation of a user interface for non-standard technologies.

b. Indexing and Retrieval

The indexed search script developed in this thesis assumes that the consumer is looking for exact matches to his query. However, there may be some technologies that are closely related to a consumer's query without being an exact match. The taxonomy for indexing technologies is already in place and functional (Rogers, 1996). A method for assigning "scores" based on a consumer's request would prove to be beneficial.

c. Distributed Processing

All of the CGI scripts and database tables for the current DecisionNet prototype reside on one machine. A system failure on this machine would be catastrophic. Further research is needed to determine a more reliable method of processing. Such methods could include the use of mirror sites or distributed processing systems.

d. Electronic Commerce

The current implementation of DecisionNet assumes that all transactions are processed at no cost to consumers or providers. Eventually, DecisionNet should operate in a manner that will allow the passing of costs on to users. Further research is needed to combine pricing schemes (Brownlee, 1996), financial transaction mechanisms (Palumbo, 1996) and advertising (Bhargava et al., June 1995) into an electronic commerce framework for DecisionNet.

2. Potential DoD Uses for this Technology

Dan McQuay's thesis (1995) describes several uses for Distributed Decision Support Networks (DDSN) and Modeling and Simulation (M & S) for the Department of Defense, including military schools, the acquisition community, and battlefield analysis. A system such as DecisionNet could be used as the central broker for decision technologies in support of DoD objectives. Although the DecisionNet system is primarily designed for the purpose of brokering transactions between consumers and providers of decision support technologies, this system could easily be adapted to any sort of database application, including personnel records, financial data, and spare parts inventories at Defense Industrial Supply Centers.

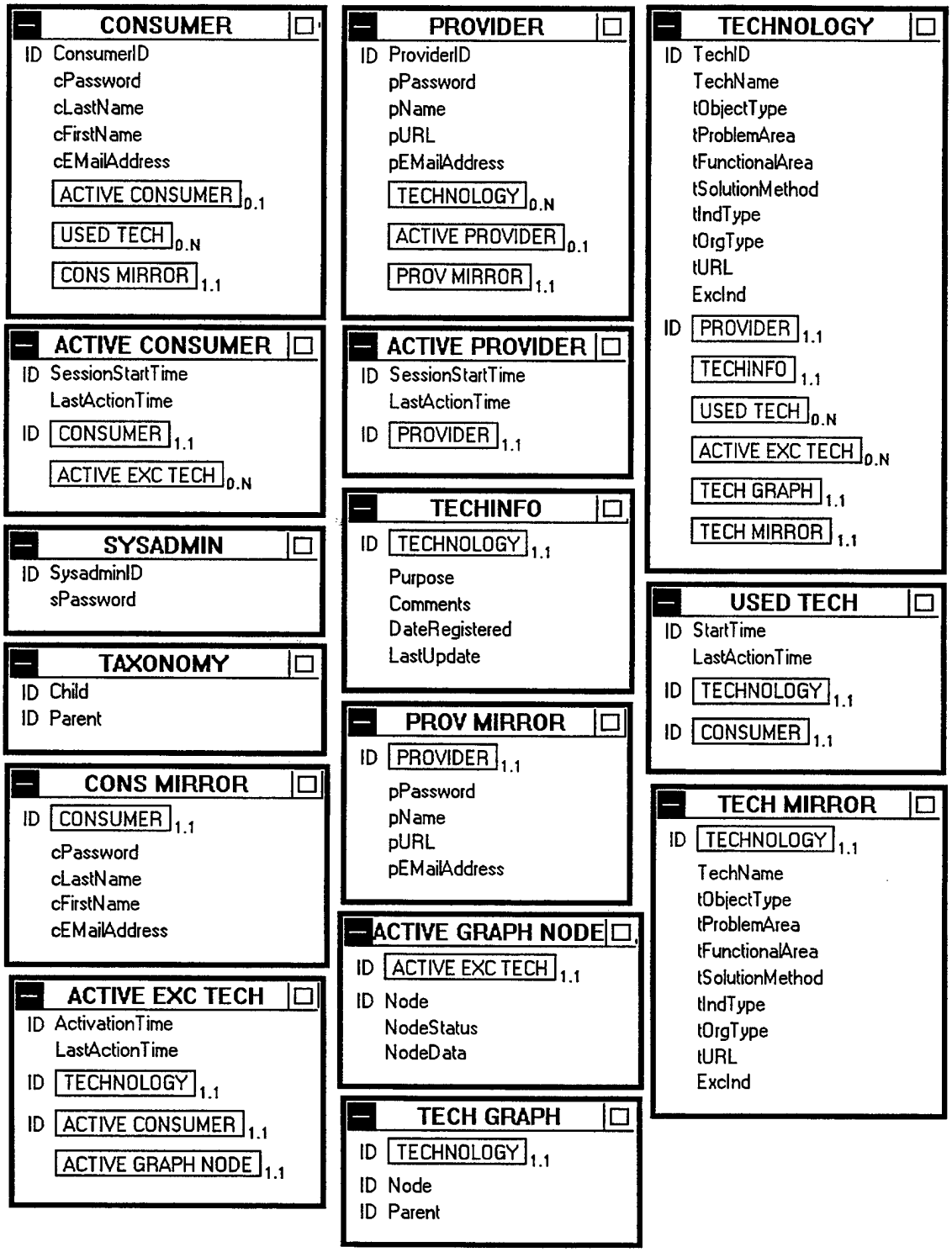
The concept of using the World Wide Web as a communications medium for seamless access to a large database system is fairly unexplored. Linking users to a highly functional DBMS, either through the WWW or corporate "Intranets," may be a viable and cost-effective alternative to stovepipe systems.

C. CLOSING REMARKS

The ultimate goal of the DecisionNet concept is to provide decision makers and researchers global access, over the Internet, to a large distributed collection of decision technologies (Bhargava et al., June 1995). The system developed through this thesis has helped to address this goal. By maintaining pertinent metadata in a database, users are provided with rapid, convenient, and powerful access to decision support information. As

the Internet continues its staggering growth, users will learn of the benefits of allowing a system such as DecisionNet to broker transactions for distributed decision support technologies.

APPENDIX A. SEMANTIC OBJECT DIAGRAM



APPENDIX B. DATA DICTIONARY

Semantic Object: **Active Consumer**
Table Name: **ACT_CONS.DB**

Attribute	Description	Key Field	Required Field	Category	Other Tables With this Attribute
ActConsID	Consumer login ID (same as ConsumerID)	Yes	Yes	General	ACTXTECH ACT_NODE CONSUMER CONSMIRR USEDTECH
SessionStartTime	Active Session Starting Time	Yes	Yes	Session	ACT_PROV
LastActionTime	Active Session Last Action Time	No	Yes	Session	ACT_PROV ACTXTECH USEDTECH

Semantic Object: **Active Exclusive Technology**
Table Name: **ACTXTECH.DB**

Attribute	Description	Key Field	Required Field	Category	Other Tables With this Attribute
ProviderID	Provider of a technology	Yes	Yes	General	ACT_NODE ACT_PROV PROVIDER PROVMIRR TECHGRAP TECHINFO TECHMIRR TECHNOLO USEDTECH
TechID	Technology serial number (by provider)	Yes	Yes	General	ACT_NODE TECHGRAP TECHINFO TECHMIRR TECHNOLO USEDTECH
ActConsID	Active Consumer using technology (same as ConsumerID)	Yes	Yes	General	ACT_CONS ACT_NODE CONSMIRR CONSUMER USEDTECH
ActivationTime	Date and time exclusive technology became active	Yes	Yes	Session	ACT_NODE
LastActionTime	Date and time of last activity for technology	No	Yes	Session	ACT_CONS ACT_PROV USEDTECH

Semantic Object: **Active Graph Node**
Table Name: **ACT_NODE.DB**

Attribute	Description	Key Field	Required Field	Category	Other Tables With this Attribute
ProviderID	Provider of a technology	Yes	Yes	General	ACT_PROV ACTXTECH PROVIDER PROVMIRR TECHGRAP TECHINFO TECHMIRR TECHNOLO USEDTECH
TechID	Technology serial number (by provider)	Yes	Yes	General	ACTXTECH TECHGRAP TECHINFO TECHMIRR TECHNOLO USEDTECH
ActConsID	Active Consumer using technology (same as ConsumerID)	Yes	Yes	General	ACT_CONS ACTXTECH CONSMIRR CONSUMER USEDTECH
ActivationTime	Date and time exclusive technology became active	Yes	Yes	Session	ACTXTECH
Node	Current Node	Yes	Yes	Session	TECHGRAP
NodeStatus	Status of Node for data entry (not ready, ready, entered)	No	Yes	Session	none
NodeData	Data for a node	No	No	Session	none

Semantic Object: **Active Provider**
Table Name: **ACT_PROV.DB**

Attribute	Description	Key Field	Required Field	Category	Other Tables With this Attribute
ActProvID	Provider login ID (same as ProviderID)	Yes	Yes	General	ACT_NODE ACTXTECH PROVIDER PROVMIRR TECHGRAP TECHINFO TECHMIRR TECHNOLO USEDTECH
SessionStartTime	Active Session Starting Time	Yes	Yes	Session	ACT_CONS
LastActionTime	Active Session Last Action Time	No	Yes	Session	ACT_CONS ACTXTECH USEDTECH

Semantic Object: **Consumer**
Table Name: **CONSUMER.DB**

Attribute	Description	Key Field	Required Field	Category	Other Tables With this Attribute
ConsumerID	Consumer login ID	Yes	Yes	General	ACT_CONS ACT_NODE ACTXTECH CONSMIRR USEDTECH
cPassword	Consumer's password	No	Yes	General	CONSMIRR
cLastName	Consumer's last name	No	No	Archival	CONSMIRR
cFirstName	Consumer's first name	No	No	Archival	CONSMIRR
cEMailAddress	Consumer's e-mail address	No	Yes	Archival	CONSMIRR

Semantic Object: **Consumer Mirror**
Table Name: **CONSMIRR.DB**

Attribute	Description	Key Field	Required Field	Category	Other Tables With this Attribute
ConsumerID	Consumer login ID	Yes	Yes	General	ACT_CONS ACT_NODE ACTXTECH CONSUMER USEDTECH
cPassword	Consumer's password	No	Yes	General	CONSUMER
cLastName	Consumer's last name	No	No	Archival	CONSUMER
cFirstName	Consumer's first name	No	No	Archival	CONSUMER
cEMailAddress	Consumer's e-mail address	No	Yes	Archival	CONSUMER

Semantic Object: **Provider**
Table Name: **PROVIDER.DB**

Attribute	Description	Key Field	Required Field	Category	Other Tables With this Attribute
ProviderID	Provider login ID	Yes	Yes	General	ACT_NODE ACT_PROV ACTXTECH PROVMIRR TECHGRAP TECHINFO TECHMIRR TECHNOLO USEDTECH
pPassword	Provider's password	No	Yes	General	PROVMIRR
pName	Provider's name (person or company)	No	Yes	General	PROVMIRR
pURL	Provider's home page URL	No	No	General	PROVMIRR
pEMailAddress	Provider's e-mail address	No	Yes	Archival	PROVMIRR

Semantic Object: **Provider Mirror**
Table Name: **PROVMIRR.DB**

Attribute	Description	Key Field	Required Field	Category	Other Tables With this Attribute
ProviderID	Provider login ID	Yes	Yes	General	ACT_NODE ACT_PROV ACTXTECH PROVIDER TECHGRAP TECHINFO TECHMIRR TECHNOLO USEDTECH
pPassword	Provider's password	No	Yes	General	PROVIDER
pName	Provider's name (person or company)	No	Yes	General	PROVIDER
pURL	Provider's home page URL	No	No	General	PROVIDER
pEMailAddress	Provider's e-mail address	No	Yes	Archival	PROVIDER

Semantic Object: **System Administrator**
Table Name: **SYSADMIN.DB**

Attribute	Description	Key Field	Required Field	Category	Other Tables With this Attribute
SysadminID	System administrator login ID	Yes	Yes	General	none
sPassword	System administrator's password	No	Yes	General	none

Semantic Object: **Taxonomy**
Table Name: **TAXONOMY.DB**

Attribute	Description	Key Field	Required Field	Category	Other Tables With this Attribute
Child	Category for indexing of technologies	Yes	Yes	Archival	none
Parent	Parent of a given child category	Yes	Yes	Archival	TECHGRAP

Semantic Object: **Technology**
Table Name: **TECHNOLO.DB**

Attribute	Description	Key Field	Required Field	Category	Other Tables With this Attribute
ProviderID	Provider of a technology	Yes	Yes	General	ACT_NODE ACT_PROV ACTXTECH PROVIDER PROVMIRR TECHGRAP TECHINFO TECHMIRR USEDTECH
TechID	Technology serial number (by provider)	Yes	Yes	General	ACT_NODE ACTXTECH TECHGRAP TECHINFO TECHMIRR USEDTECH
TechName	Technology name	No	Yes	General	TECHMIRR
tObjectType	Object Type	No	Yes	Archival	TECHMIRR
tProblemArea	Problem Area	No	Yes	Archival	TECHMIRR
tFunctionalArea	Functional Area	No	Yes	Archival	TECHMIRR
tSolutionMethod	Solution Method	No	Yes	Archival	TECHMIRR
tIndType	Industry Type	No	Yes	Archival	TECHMIRR
tOrgType	Organization Type	No	Yes	Archival	TECHMIRR
tURL	URL of executable technology	No	Yes	General	TECHMIRR
ExcInd	Exclusive or Independent	No	Yes	General	TECHMIRR

Semantic Object: **Technology Graph**
Table Name: **TECHGRAP.DB**

Attribute	Description	Key Field	Required Field	Category	Other Tables With this Attribute
ProviderID	Provider of a technology	Yes	Yes	General	ACT_NODE ACT_PROV ACTXTECH PROVIDER PROVMIRR TECHINFO TECHMIRR TECHNOLO USEDTECH
TechID	Technology serial number (by provider)	Yes	Yes	General	ACT_NODE ACTXTECH TECHINFO TECHMIRR TECHNOLO USEDTECH
Node	Node for data entry	Yes	Yes	Archival	ACT_NODE
Parent	Parent of a given node	Yes	Yes	Archival	TAXONOMY

Semantic Object: **Technology Information**
Table Name: **TECHINFO.DB**

Attribute	Description	Key Field	Required Field	Category	Other Tables With this Attribute
ProviderID	Provider of a technology	Yes	Yes	General	ACT_NODE ACT_PROV ACTXTECH PROVIDER PROVMIRR TECHGRAP TECHMIRR TECHNOLO USEDTECH
TechID	Technology serial number (by provider)	Yes	Yes	General	ACT_NODE ACTXTECH TECHGRAP TECHMIRR TECHNOLO USEDTECH
Purpose	Technology's main function	No	No	Archival	none
Comments	Special instructions or comments	No	No	Archival	none
DateRegistered	Date Technology registered into DecisionNet	No	Yes	Archival	none
LastUpdate	Date of last update to technology registration data	No	Yes	Archival	none

Semantic Object: **Technology Mirror**
Table Name: **TECHMIRR.DB**

Attribute	Description	Key Field	Required Field	Category	Other Tables With this Attribute
ProviderID	Provider of a technology	Yes	Yes	General	ACT_NODE ACT_PROV ACTXTECH PROVIDER PROVMIRR TECHGRAP TECHINFO TECHNOLO USEDTECH
TechID	Technology serial number (by provider)	Yes	Yes	General	ACT_NODE ACTXTECH TECHGRAP TECHINFO TECHNOLO USEDTECH
TechName	Technology name	No	Yes	General	TECHNOLO
tObjectType	Object Type	No	Yes	Archival	TECHNOLO
tProblemArea	Problem Area	No	Yes	Archival	TECHNOLO
tFunctionalArea	Functional Area	No	Yes	Archival	TECHNOLO
tSolutionMethod	Solution Method	No	Yes	Archival	TECHNOLO
tIndType	Industry Type	No	Yes	Archival	TECHNOLO
tOrgType	Organization Type	No	Yes	Archival	TECHNOLO
tURL	URL of executable technology	No	Yes	General	TECHNOLO
ExcInd	Exclusive or Independent	No	Yes	General	TECHNOLO

Semantic Object: **Used Technology**
Table Name: **USEDTECH.DB**

Attribute	Description	Key Field	Required Field	Category	Other Tables With this Attribute
ProviderID	Provider of a technology	Yes	Yes	General	ACT_NODE ACT_PROV ACTXTECH PROVIDER PROVMIRR TECHGRAP TECHINFO TECHMIRR TECHNOLO
TechID	Technology serial number (by provider)	Yes	Yes	General	ACT_NODE ACTXTECH TECHGRAP TECHINFO TECHMIRR TECHNOLO
ConsumerID	Consumer who used a technology	No	Yes	Archival	ACT_CONS ACT_NODE ACTXTECH CONSMIRR CONSUMER
StartTime	Date and time technology began to be used	No	Yes	Session	none
LastActionTime	Date and time of last activity with technology	No	Yes	Session	ACT_CONS ACT_PROV ACTXTECH

APPENDIX C. CGI SCRIPTS

A. GENERAL USER SCRIPTS

1. Browse DecisionNet Technologies

```
unit Browse1;   {DNet Browse for unregistered user; created by Steve Earley,
                 last update 20 May 96.}

interface

uses
 SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
 Forms, Dialogs, DB, Cgidb, Cgi, DBTables;

type
 TForm1 = class(TForm)
  CGIEnvData1: TCGIEnvData;
  CGIDB1: TCGIDB;
  DataSource1: TDataSource;
  Query1: TQuery;
  procedure FormCreate(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form1: TForm1;

implementation
{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin
   with CGIEnvData1 do
   begin
      websiteINIFilename := paramstr(1);
      application.onException := cgiErrorHandler;
      application.processMessages;
```

```
createStdout;
sendPrologue;

send( '<HTML><HEAD>' );
sendTitle( 'List of Technologies' );
send( '</HEAD><BODY BGCOLOR="80B7B0">' );
send('<center><H1>DecisionNet Technologies</h1></center>');
send('This is a complete listing of all registered technologies. ');
send(' The technologies are owned and maintained by their ');
send('individual providers.  DecisionNet"s purpose is to ');
send('facilitate access to these technologies. ');
send(' If you want to access these programs, you must first ');
send('<A HREF="http://131.120.39.63/dnet/consumer/register.exe">');
send('register</A> as a consumer. ');
send('<p>');

send('<center>');

with query1 do
 begin
 close;
 SQL.clear;
 SQL.add('SELECT TECHNOLO."TechName", PROVIDER."pName", ' +
       'TECHNOLO."tObjectType", TECHNOLO."tProblemArea" ' +
       'FROM TECHNOLO,PROVIDER ' +
       'WHERE (TECHNOLO.ProviderID = PROVIDER.ProviderID)');

 open;
 fieldByName ( 'TechName' ).displayLabel := 'Technology Name' ;
 fieldByName ( 'pName' ).displayLabel := 'Provider Name' ;
 fieldByName ( 'tObjectType' ).displayLabel := 'Object Type' ;
 fieldByName ( 'tProblemArea' ).displayLabel := 'Problem Area' ;
end;

CGIDB1.drawTable;
query1.close;

send('</center>');
send('<p>');

send('<CENTER> ' +
         '<TABLE BORDER=6 CELLPADDING=6> ' +
         '<TR ALIGN="CENTER" VALIGN=MIDDLE> ' +
```

```
            '<TD ALIGN="CENTER" VALIGN=MIDDLE> ' +
            '<A HREF="http://dnet.sm.nps.navy.mil/consumer/register.htm"> ' +
            'Register as Consumer</A></TD>');
        send('<TD ALIGN="CENTER" VALIGN=MIDDLE> ' +
            '<A HREF="http://dnet.sm.nps.navy.mil/provider/register.htm"> ' +
            'Register as Provider</A></TD>');
        send('<TD ALIGN="CENTER" VALIGN=MIDDLE> ' +
            '<A HREF="http://dnet.sm.nps.navy.mil/welcome.htm"> ' +
            'Return to Welcome Page</A></TD>');
        send('</TR></TABLE></CENTER>');

        send('<p>');

        sendHR;

        send('<p><I><FONT SIZE=-1>');
         send( 'This application was created by Steve Earley ');
        send( 'for Professor Hemant Bhargava.<br>' );
        send( 'Generated on ' + webdate(now) );

        send( '</FONT></I></BODY></HTML>' );

        closeStdout;
         closeApp( application );
    end;
end;
end.
```

2. Null Script

```
unit Null1;     {General User Null Script; Created by Steve Earley.
                Last Update 23 April 1996.}

interface

uses
 SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
 Forms, Dialogs, Cgi;

type
 TForm1 = class(TForm)
  CGIEnvData1: TCGIEnvData;
  procedure FormCreate(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin
   with CGIEnvData1 do
    begin
      websiteINIFilename := paramstr(1);
      application.onException := cgiErrorHandler;
      application.processMessages;
      createStdout;
      bounceToLocation('http://dnet.sm.nps.navy.mil/null.htm');
      closeApp( application );
    end;
end;
end.
```

B. CONSUMER SCRIPTS

1. Consumer Registration

```
unit Registe1; {Consumer Registration; created by Steve Earley;
               last update 10 Jun 96.}

interface

uses
 SysUtils, WinTypes, WinProcs, Messages, Classes,
 Forms, Cgi, DB, DBTables;

type
 TForm1 = class(TForm)
  DataSource1: TDataSource;
  Table1: TTable;
  CGIEnvData1: TCGIEnvData;
  DataSource2: TDataSource;
  Table2: TTable;
  DataSource3: TDataSource;
  Table3: TTable;
  procedure FormCreate(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);

var
 ConsumerID : string;
 cPassword : string;
 cPassword2 : string;
 cLastName : string;
```

```
  cFirstName : string;
  cEMailAddress : string;

begin
  with CGIEnvData1 do
  begin
    {required when this program runs under WebSite}
    webSiteINIFilename := paramstr(1);
    application.onException := cgiErrorHandler;
    application.processMessages;

    {receive input fields from HTML form}
    ConsumerID := getSmallField( 'ConsumerID' );
    cPassword := getSmallField( 'cPassword' );
    cPassword2 := getSmallField( 'cPassword2' );
    cLastName := getSmallField( 'cLastName' );
    cFirstName := getSmallField( 'cFirstName' );
    cEMailAddress := getSmallField( 'cEMailAddress' );

    {HTML page header info}
    createStdout;
    sendPrologue;
    send( '<HTML><HEAD>' );
    sendTitle( 'DecisionNet Consumer Registration' );
    send( '</HEAD><BODY BGCOLOR="80B7B0">' );

    with Table1 do    {puts cursor on correct record in Consumer table}
      begin
        open;
        SetKey;
        FieldByName('ConsumerID').AsString := ConsumerID;
        GotoKey;
      end;

    with Table2 do    {puts cursor on correct record in Consumer Mirror table}
      begin
        open;
        SetKey;
        FieldByName('ConsumerID').AsString := ConsumerID;
        GotoKey;
      end;

    if (Table1.GoToKey = True) or (Table2.GoToKey = True)
```

```
  or (ConsumerID = CGINotFound) then

  {User's ID choice already exists on Consumer or ConsMirror table,
  or no ConsumerID entered on form}
  begin
    send('<center><h1>Invalid Consumer ID</h1>');
    send('<h2>Sorry, but you cannot use the Consumer ID you have ');
    send('chosen.  Please try another one.</h2></center>');
    send('<p>');
    sendHR;
    send('<p>');

     send('<CENTER> ' +
     '<TABLE BORDER=6 CELLPADDING=6> ' +
     '<TR ALIGN="CENTER" VALIGN=MIDDLE> ' +
       '<TD ALIGN="CENTER" VALIGN=MIDDLE> ' +
       '<A HREF="http://dnet.sm.nps.navy.mil/consumer/register.htm"> ' +
     'Return to Consumer Registration</A></TD>');
     send('</TR></TABLE></CENTER>');

     send('<p>');
  end

else {valid Consumer ID chosen}
  begin
    if cPassword <> cPassword2 then {bad registration}
     begin
      send('<center><h1>Password Mismatch</h1>');
      send('<h2>Please verify your password choice, and try ');
      send('again.</h2></center>');
      send('<p>');
      sendHR;
      send('<p>');

      send('<CENTER> ' +
      '<TABLE BORDER=6 CELLPADDING=6> ' +
      '<TR ALIGN="CENTER" VALIGN=MIDDLE> ' +
        '<TD ALIGN="CENTER" VALIGN=MIDDLE> ' +
        '<A HREF="http://dnet.sm.nps.navy.mil/consumer/register.htm"> ' +
      'Return to Consumer Registration</A></TD>');
      send('</TR></TABLE></CENTER>');
      send('<p>');
     end
```

```
    else {good registration}
      begin
        {input record into Consumer table}
        Table1.AppendRecord([ConsumerID, cPassword, cLastName,
              cFirstName, cEMailAddress]);
        Table1.close;

        {input record into Consumer Mirror table}
        Table2.AppendRecord([ConsumerID, cPassword, cLastName,
              cFirstName, cEMailAddress]);
        Table2.close;

        {input record into Active Consumer table}
        Table3.open;
        Table3.AppendRecord([ConsumerID, DateTimeToStr(Now),
              DateTimeToStr(Now)]);
        Table3.close;

        send('<center><h1>Welcome to DecisionNet!</h1>');
        send('<h2>You are now registered under the ' +
          'User Name <I>' + ConsumerID  + '</I>. ');
        send(' Thank you for using DecisionNet.</h2></center>');
        send('<p>');
        sendHR;
        send('<p>');

        {capture ConsumerID, send user to Consumer Menu}
        send('<CENTER>');
        send('<FORM method=post action="'+
          'http://131.120.39.63/cgi-win/dnet/consumer/menu.exe">');
        send('<input type="hidden" name="ConsumerID" '+
          'value="'+ConsumerID+'"></td>');
        send('<input type="submit" value="Registered Consumer Menu">');
        send('</form>');
        send('</CENTER>');

        send('<p>');
      end;
  end;

{HTML page footer info}
send('<HR>');
send( 'This application was created by Steve Earley for Professor ');
```

```
    send( 'Hemant Bhargava.<br>' );
    send( 'Generated on ' + webdate(now) );
    send( '</BODY></HTML>' );
    closeStdout;
  end;
end;
end.
```

2. Consumer Login Script

```
unit Login1; {Consumer Login; created by Steve Earley; updated: 13 May 96.}

interface

uses
 SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
 Forms, Dialogs, Cgi, DBTables, DB;

type
 TForm1 = class(TForm)
  DataSource1: TDataSource;
  DataSource2: TDataSource;
  Table1: TTable;
  Table2: TTable;
  CGIEnvData1: TCGIEnvData;
  procedure FormCreate(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);

var
 ConsumerID : string;
 cPassword : string;

begin
 with CGIEnvData1 do
 begin
   { required when this program runs under WebSite}
   webSiteINIFilename := paramstr(1);
   application.onException := cgiErrorHandler;
```

```
application.processMessages;

{input fields from Login form (start.htm)}
ConsumerID := getSmallField( 'ConsumerID' );
cPassword := getSmallField( 'cPassword' );

{standard dynamic HTML header information}
createStdout;
sendPrologue;
send( '<HTML><HEAD>' );
sendTitle( 'DecisionNet Consumer Login' );
send( '</HEAD><BODY BGCOLOR="80B7B0">' );


with Table1 do   {puts cursor on correct record in Consumer table;
            GotoKey returns True if record is valid}
   begin
      open;
      SetKey;
      FieldByName('ConsumerID').AsString := ConsumerID;
      GotoKey;
   end;

if (Table1.FieldByName('cPassword').AsString <> cPassword) or
 (Table1.GoToKey = False) then   {password does not match ConsumerID, or
                      user not in Consumer table}

  begin
     send('<center><h1>Incorrect Login!</h1></center>');
     send('<h2>Please verify that your User Name and Password ' +
        'are correct, then try again.</h2>');
     send('<p>');
     sendHR;
     send('<p>');

     send('<CENTER> ' +
        '<TABLE BORDER=6 CELLPADDING=6> ' +
        '<TR ALIGN="CENTER" VALIGN=MIDDLE> ' +
          '<TD ALIGN="CENTER" VALIGN=MIDDLE> ' +
          '<A HREF="http://dnet.sm.nps.navy.mil/start.htm"> ' +
        'DecisionNet Start Page</A></TD>');
     send('</TR></TABLE></CENTER>');
  end
```

```
else  {ConsumerID matches cPassword}
  begin
     with Table2 do   {search for user in Active Consumer table;
                  GotoKey returns True if already logged in}
       begin
         open;
         SetKey;
         FieldByName('ActConsID').AsString := ConsumerID;
         GoToKey;
       end;

       if Table2.GoToKey = True then   {already logged in}
         begin
           send('<center><h1>Already logged in</h1>');
           send('<h2>You were previously logged in under the ');
           send('User Name <I>' + ConsumerID  + '</I> ');
           send('and did not logout. Your last action was at ');
           send(Table2.FieldByName('LastActionTime').AsString +'.');
           send(' There is no need to login again.');
           send('<P>');
           send(' Thank you for using DecisionNet.</h2></center>');
           send('<p>');
         end
       else
         begin
           {put user in Active Consumer table}
           with Table2 do
           begin
           open;
           AppendRecord([ConsumerID, DateTimeToStr(Now),
             DateTimeToStr(Now)]);
           end;

           send('<center><h1>Welcome to DecisionNet!</h1>');
           send('<h2>You are logged in under the ' +
             'User Name <I>' + ConsumerID  + '</I>. ');
           send(' Thank you for using DecisionNet.</h2></center>');
           send('<p>');
           sendHR;
           send('<p>');
         end;
```

```
        {capture ConsumerID, send user to Consumer Menu}
        send('<CENTER>');
        send('<FORM method=post action="'+
              'http://131.120.39.63/cgi-win/dnet/consumer/menu.exe">');
        send('<input type="hidden" name="ConsumerID" '+
              'value="'+ConsumerID+'"></td>');
        send('<input type="submit" value="Registered Consumer Menu">');
        send('</form>');
        send('</CENTER>');
      end;

    Table1.close;
    Table2.close;

    {standard dynamic HTML footer information}
    send('<p>');
    sendHR;
    send('<p><i><font size=-1>');
    send( 'This application was created by Steve Earley ');
    send( 'for Professor Hemant Bhargava.<br>' );
    send( 'Generated on ' + webdate(now) );
    send( '</i></font></BODY></HTML>' );
    closeStdout;

  end;
end;
end.
```

3. Consumer Menu

```
unit Menu1;   {Active Consumer menu; created by Steve Earley,
              Last updated 20 Jun 96}

interface

uses

 SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
 Forms, Dialogs, DB, Cgidb, Cgi, DBTables;

type
 TForm1 = class(TForm)
  CGIEnvData1: TCGIEnvData;
  Query1: TQuery;
  DataSource1: TDataSource;
  DataSource2: TDataSource;
  DataSource3: TDataSource;
  Table1: TTable;
  Query2: TQuery;
  DataSource4: TDataSource;
  Query3: TQuery;
  procedure FormCreate(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);

var
  ConsumerID: string;

begin
```

```
with CGIEnvData1 do
 begin
  websiteINIFilename := paramstr(1);
  application.onException := cgiErrorHandler;
  application.processMessages;
  createStdout;
  sendPrologue;

  {standard header information}
  send( '<HTML><HEAD>' );
  sendTitle( 'DecisionNet Consumer Menu' );
  send( '</HEAD><BODY BGCOLOR="80B7B0">' );

  {Get ConsumerID from start page; if user tries to go directly to
  menu, raise an error}
  ConsumerID := GetSmallField('ConsumerID');

  if ConsumerID = CGINotFound then
   begin
      send('<center><H1>Not Logged In</h1>');
      send('<h2>You are currently not logged in to DecisionNet. ');
      send('Please <a href="http://131.120.39.66/start.htm">login</a> ');
      send('to continue.</h2></center>');
   end

  else
   begin

    {update Active Consumer table}
    with Query1 do
          begin
             close;
             SQL.clear;
             sql.add('UPDATE ACT_CONS ');
             sql.add('SET LastActionTime = ''' +DateTimeToStr(Now)+ '''');
             sql.add('WHERE ActConsID = ''' + ConsumerID + '''');
             ExecSQL;
          end;

    send('<center><H1>DecisionNet Consumer Menu</h1></center>');

    {List Technologies form}
    send('<H3>List Technologies</H3>');
```

```
send('Obtain a listing of all technologies sorted by: ');
send('<CENTER>');
send('<FORM METHOD=POST ACTION= ');
send('"http://131.120.39.63/cgi-win/dnet/consumer/listtech.exe">');
send('<INPUT TYPE=HIDDEN NAME="ConsumerID" VALUE="'+ConsumerID+'">');
send('<SELECT NAME="sortkey" TYPE = "text" SIZE=1>');
send('<OPTION SELECTED VALUE="TechName">Technology Name');
send('<OPTION VALUE="pName">Provider Name');
send('<OPTION VALUE="tObjectType">Object Type');
send('<OPTION VALUE="tProblemArea">Problem Type');
send('</SELECT><P>');
send('<INPUT TYPE=SUBMIT VALUE="List Technologies" ALIGN="MIDDLE">');
send('</CENTER></FORM>');

sendHR;

{Access Technology form -- calls the "about" script to launch technology}
send('<H3>Access Technology</H3>');
send('If you already know the Provider Name and TechID Number of the ');
send('technology you wish to use, please choose them.');

send('<center>');
send('<TABLE BORDER=6 CELLPADDING=6>');
send('<FORM METHOD = POST ACTION= ');
send('"http://131.120.39.63/cgi-win/dnet/consumer/about.exe">');
send('<INPUT TYPE=HIDDEN NAME="ConsumerID" VALUE="'+ConsumerID+'">');

send('<TR>');
send('<TD>Provider Name:</TD>');
send('<TD><SELECT NAME="ProviderID" TYPE="text" SIZE=1>');

with Query2 do    {Use a query of Technolo.db and provider.db to list
                   Providers, capture ProviderID for each option}
 begin
  close;
  SQL.Add('SELECT DISTINCT TECHNOLO.ProviderID, PROVIDER.pName ');
  SQL.Add('FROM TECHNOLO, PROVIDER ');
  SQL.Add('WHERE TECHNOLO.ProviderID=PROVIDER.ProviderID ');
  SQL.Add('ORDER BY pName');
  open;
```

```
   first; {puts cursor on first record in answer table}
   while not EOF do
    begin
       send('<OPTION VALUE= "'+FieldByName('ProviderID').AsString+'">');
       send(FieldByName('pName').AsString);
       next; {puts cursor on next record}
    end;
   close;
  end;
send('</SELECT></TD></TR>');

send('<TR>');
send('<TD>TechID Number:</TD>');
send('<TD><SELECT NAME="TechID" TYPE="text" SIZE=1>');

with Query3 do    {Use a query of Technolo.db to list TechID numbers,
                   capture TechID for each option}
 begin
   close;
   SQL.Add('SELECT DISTINCT TechID ');
   SQL.Add('FROM TECHNOLO ');
   open;

   first; {puts cursor on first record in answer table}
   while not EOF do
    begin
       send('<OPTION VALUE= "'+FieldByName('TechID').AsString+'">');
       send(FieldByName('TechID').AsString);
       next; {puts cursor on next record}
    end;
   close;
  end;

send('</SELECT></TD></TR>');
send('<TR ALIGN="CENTER" VALIGN="MIDDLE"><TD COLSPAN=2>');
send('<INPUT TYPE=SUBMIT VALUE="Access Technology" ALIGN="MIDDLE">');
send('</TD></FORM></TABLE>');
send('</center>');
sendHR;

{Indexed Search form}
send('<h3>Indexed Search</h3>');
send('DecisionNet technologies are classified along six dimensions. ');
```

```
send('Define your search criteria by choosing terms from each.');
send('<center>');
send('<FORM METHOD=POST ACTION= ');
send('"http://131.120.39.63/cgi-win/dnet/consumer/indexed.exe">');
send('<INPUT TYPE=HIDDEN NAME="ConsumerID" VALUE="'+ConsumerID+'">');

send('<TABLE BORDER=6 CELLPADDING=6>');
send('<TR><TD>Object Type:');
send('</td><td><SELECT NAME="tObjectType" TYPE="text" SIZE=1>');
send('<OPTION SELECTED VALUE="ALL">ALL');

with Table1 do    {Check Taxonomy (master.db) for Option Entries}
 begin
  open;
  first; {puts cursor on first record in table}

  while not EOF do
   begin
    if FieldByName('Parent').AsString = 'Object Type' then
     begin
      send('<OPTION VALUE= "'+FieldByName('Child').AsString+'">');
      send(FieldByName('Child').AsString);
     end;
      next; {puts cursor on next record}
   end;
  close;
 end;

send('</SELECT></TD>');
send('<TR><TD>Problem Area:');
send('</TD> <td> <SELECT NAME="tProblemArea" SIZE=1>');
send('<OPTION SELECTED VALUE="ALL">ALL');

with Table1 do    {Check Taxonomy (master.db) for Option Entries}
 begin
  open;
  first; {puts cursor on first record in table}

  while not EOF do
   begin
    if FieldByName('Parent').AsString = 'Problem Area' then
       begin
       send('<OPTION VALUE= "'+FieldByName('Child').AsString+'">');
```

```
        send(FieldByName('Child').AsString);
        end;
      next; {puts cursor on next record}
    end;
  close;
 end;
send('</SELECT></TD>');

send('<TR><TD>Functional Area:');
send('</td><td><SELECT NAME="tFunctionalArea" TYPE="text" SIZE=1>');
send('<OPTION SELECTED VALUE="ALL">ALL');

with Table1 do    {Check Taxonomy (master.db) for Option Entries}
 begin
  open;
  first; {puts cursor on first record in table}

  while not EOF do
   begin
    if FieldByName('Parent').AsString = 'Functional Area' then
      begin
      send('<OPTION VALUE= "'+FieldByName('Child').AsString+'">');
      send(FieldByName('Child').AsString);
      end;
    next; {puts cursor on next record}
   end;
  close;
 end;
send('</SELECT></TD>');

send('<TR><TD>Solution Method:');
send('</TD> <td><SELECT NAME="tSolutionMethod" SIZE=1>');
send('<OPTION SELECTED VALUE="ALL">ALL');

with Table1 do    {Check Taxonomy (master.db) for Option Entries}
 begin
  open;
  first; {puts cursor on first record in table}

  while not EOF do
   begin
    if FieldByName('Parent').AsString = 'Solution Method' then
      begin
```

```
          send('<OPTION VALUE= "'+FieldByName('Child').AsString+'">');
          send(FieldByName('Child').AsString);
         end;
       next; {puts cursor on next record}
     end;
    close;
  end;
send('</SELECT></TD>');

send('<TR><TD>Industry Type:');
send('</TD> <td><SELECT NAME="tIndType" SIZE=1>');
send('<OPTION SELECTED VALUE="ALL">ALL');

with Table1 do    {Check Taxonomy (master.db) for Option Entries}
  begin
   open;
   first; {puts cursor on first record in table}

   while not EOF do
    begin
     if FieldByName('Parent').AsString = 'Industry Type' then
        begin
        send('<OPTION VALUE= "'+FieldByName('Child').AsString+'">');
        send(FieldByName('Child').AsString);
        end;
      next; {puts cursor on next record}
    end;
   close;
  end;
send('</SELECT></TD>');

send('<tr><td>Organization Type:');
send('</td><td><SELECT NAME="tOrgType" SIZE=1>');
send('<OPTION SELECTED VALUE="ALL">ALL');

with Table1 do    {Check Taxonomy (master.db) for Option Entries}
  begin
   open;
   first; {puts cursor on first record in table}

   while not EOF do
    begin
     if FieldByName('Parent').AsString = 'Organization Type' then
```

```
            begin
            send('<OPTION VALUE= "'+FieldByName('Child').AsString+'">');
            send(FieldByName('Child').AsString);
            end;
          next; {puts cursor on next record}
        end;
      close;
    end;
  send('</SELECT></TD>');

  send('<TR ALIGN="CENTER"><TD COLSPAN=2>');
send('<INPUT TYPE=SUBMIT VALUE="Find Technologies"'+
      ' ALIGN="MIDDLE"></TD>');
  send('</TR>');
  send('</TABLE>');
  send('</FORM>');
  send('</CENTER>');
  sendHR;

  {Keyword Search form}
  send('<H3>Keyword Search');
  send('<FONT SIZE=-2>(Not yet available)</FONT></H3>');
  send('Find suitable technologies whose title/description ');
  send('contains your chosen keywords.');

  send('<CENTER>');
  send('<FORM METHOD=POST ACTION= ');
  send('"http://131.120.39.63/cgi-win/dnet/consumer/null.exe">');
  send('<INPUT TYPE=HIDDEN NAME="ConsumerID" VALUE="'+ConsumerID+'">');
  send('<INPUT TYPE=TEXT NAME="keyword" SIZE=30 MAXLENGTH=30 ');
  send('ALIGN="MIDDLE"><P>');
  send('<INPUT TYPE=SUBMIT VALUE="Submit Search" ALIGN="MIDDLE">');
  send('</FORM>');
  send('</CENTER>');
  sendHR;

  {other links available to consumer}
  send('<CENTER> ');
  send('<TABLE BORDER=6 CELLPADDING=6> ');
  send('<TR ALIGN="CENTER" VALIGN=MIDDLE> ');
  send('<FORM METHOD = POST  ACTION= ');
  send('"http://131.120.39.63/cgi-win/dnet/consumer/modify.exe">');
  send('<INPUT TYPE=HIDDEN NAME="ConsumerID" VALUE="'+ConsumerID+'">');
```

```
    send('<TD ALIGN="CENTER" VALIGN=MIDDLE> ');
    send('<INPUT TYPE=SUBMIT VALUE="Modify Info"></TD></FORM>');

    send('<FORM METHOD = POST  ACTION= ');
    send('"http://131.120.39.63/cgi-win/dnet/consumer/null.exe">');
    send('<INPUT TYPE=HIDDEN NAME="ConsumerID" VALUE="'+ConsumerID+'">');
    send('<TD ALIGN="CENTER" VALIGN=MIDDLE> ');
    send('<INPUT TYPE=SUBMIT VALUE="Account Info"></TD></FORM>');

    send('<FORM METHOD = POST  ACTION= ');
    send('"http://131.120.39.63/cgi-win/dnet/consumer/withdraw.exe">');
    send('<INPUT TYPE=HIDDEN NAME="ConsumerID" VALUE="'+ConsumerID+'">');
    send('<TD ALIGN="CENTER" VALIGN=MIDDLE> ');
     send('<INPUT TYPE=SUBMIT VALUE="Withdraw from DNet"></TD></FORM>');

    send('<FORM METHOD = POST  ACTION= ');
    send('"http://131.120.39.63/cgi-win/dnet/consumer/logout.exe">');
    send('<INPUT TYPE=HIDDEN NAME="ConsumerID" VALUE="'+ConsumerID+'">');
    send('<TD ALIGN="CENTER" VALIGN=MIDDLE> ');
    send('<INPUT TYPE=SUBMIT VALUE="Logout"></TD></FORM>');

    send('</TR></TABLE></CENTER>');
    send('<P>');

   end;
   sendHR;

   {standard footer information}
   send('<p><I><FONT SIZE=-1>');
   send( 'This application was created by Steve Earley ');
   send( 'for Professor Hemant Bhargava.<br>' );
   send( 'Generated on ' + webdate(now) );
   send( '</FONT></I></BODY></HTML>' );
   closeStdout;
   closeApp( application );

  end;
 end;
end.
```

4. List Technologies

```
unit Listtec1; {Consumer Technology Listing; created by Steve Earley.
               Last updated 29 May 96.}

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, DB, Cgidb, Cgi, DBTables;

type
  TForm1 = class(TForm)
    CGIEnvData1: TCGIEnvData;
    DataSource1: TDataSource;
    Query1: TQuery;
    DataSource2: TDataSource;
    Query2: TQuery;
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}


procedure TForm1.FormCreate(Sender: TObject);

var
   sortkey : string;
   ConsumerID : string;

begin
     with CGIEnvData1 do
     begin
        websiteINIFilename := paramstr(1);
```

```
application.onException := cgiErrorHandler;
application.processMessages;

sortkey := GetSmallField('sortkey');
ConsumerID := GetSmallField('ConsumerID');

{update Active Consumer table}
with Query2 do
   begin
      close;
      SQL.clear;
      sql.add('UPDATE ACT_CONS ');
      sql.add('SET LastActionTime = ''' +DateTimeToStr(Now)+ '''');
      sql.add('WHERE ActConsID = ''' + ConsumerID + '''');
      ExecSQL;
   end;

createStdout;
sendPrologue;

{standard header information}
send( '<HTML><HEAD>' );
sendTitle( 'List of Technologies' );
send( '</HEAD><BODY BGCOLOR="80B7B0">' );
send('<center><H1>DecisionNet Technologies</h1></center>');
send('These technologies are owned and maintained by their ');
send('individual providers.  DecisionNet''s purpose is to ');
send('facilitate access to these technologies.');
send('<p>');
send('<center>');

{build table of technologies using join of Technology and Provider}
with query1 do
 begin
    close;
    SQL.clear;
    SQL.add('SELECT TECHNOLO."TechName", PROVIDER."pName", ' +
        'TECHNOLO."tObjectType", TECHNOLO."tProblemArea", ' +
        'TECHNOLO."tURL", TECHNOLO."TechID", PROVIDER."ProviderID" ' +
        'FROM TECHNOLO,PROVIDER ' +
        'WHERE (TECHNOLO.ProviderID = PROVIDER.ProviderID)');
    SQL.add('ORDER BY '+sortkey+'');
```

```
        open;
        first; {puts cursor on first record in table}

        send('<TABLE BORDER=6 CELLPADDING=6>');
        send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
        send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
        send('<B>Technology Name</B>');
        send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
        send('<B>Provider Name</B>');
        send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
        send('<B>Object Type</B>');
        send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
        send('<B>Problem Area</B>');

        while not EOF do
         begin
          send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
          send('<FORM METHOD = POST  ACTION= ');
          send('"http://131.120.39.63/cgi-win/dnet/consumer/about.exe">');
   send('<INPUT TYPE=HIDDEN NAME="ConsumerID" VALUE="'+ConsumerID+'">');
          send('<INPUT TYPE=HIDDEN NAME="ProviderID" ');
          send('VALUE="'+ FieldByName('ProviderID').AsString +'">');
          send('<INPUT TYPE=HIDDEN NAME="TechID" ');
          send('VALUE="'+ FieldByName('TechID').AsString +'">');
          send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
        send('<INPUT TYPE=SUBMIT VALUE="'+ FieldbyName('TechName').AsString
+'">');
          send('</TD></FORM>');
          send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
          send(FieldByName('pName').AsString);
          send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
          send(FieldByName('tObjectType').AsString);
          send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
          send(FieldByName('tProblemArea').AsString);
          next; {puts cursor on next record}
         end;
        send('</TABLE>');
        close;
    end;

   send('</center>');
   send('<p>');
   sendHR;
```

```
    send('<p>');

    {capture UserID, send user to Consumer Menu}
    send('<CENTER>');
    send('<FORM method=post action="'+
     'http://131.120.39.63/cgi-win/dnet/consumer/menu.exe">');
    send('<input type="hidden" name="ConsumerID" '+
     'value="'+ConsumerID+'"></td>');
    send('<input type="submit" value="Registered Consumer Menu">');
    send('</form>');
    send('</CENTER>');

    {standard HTML footer information}
     send('<p>');
     sendHR;
     send('<p><FONT SIZE=-1><I>');
     send( 'This application was created by Steve Earley');
     send( 'for Professor Hemant Bhargava.<br>' );
     send( '<P>' );
     send( 'Generated on ' + webdate(now) );
     send( '</I></FONT></BODY></HTML>' );

     closeStdout;
     closeApp( application );

     end;
end;
end.
```

5. About Technologies Script

```
unit About1;  {Consumer "About Technology" script; created by Steve Earley.
              Last updated 18 Jun 96.}

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, DB, Cgidb, Cgi, DBTables;

type
  TForm1 = class(TForm)
    CGIEnvData1: TCGIEnvData;
    DataSource1: TDataSource;
    Query1: TQuery;
    DataSource2: TDataSource;
    Query2: TQuery;
    CGIDB1: TCGIDB;
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}


procedure TForm1.FormCreate(Sender: TObject);

var
    ConsumerID : string;   {Person using the program}
    ProviderID : string;   {Provider of a given technology}
    TechID : string;       {TechID of a given technology}

begin
```

```
with CGIEnvData1 do
  begin
   websiteINIFilename := paramstr(1);
   application.onException := cgiErrorHandler;
   application.processMessages;

   createStdout;
   sendPrologue;

   {standard header information}
   send( '<HTML><HEAD>' );
   sendTitle( 'DecisionNet Technology Details' );
   send( '</HEAD><BODY BGCOLOR="80B7B0">' );

   {Get hidden fields from previous page}
   ConsumerID := GetSmallField('ConsumerID');
   ProviderID := GetSmallField('ProviderID');
   TechID := GetSmallField('TechID');

   {update Active Consumer table}
   with Query1 do
    begin
     close;
     SQL.clear;
     sql.add('UPDATE ACT_CONS ');
     sql.add('SET LastActionTime = "' +DateTimeToStr(Now)+ '"');
     sql.add('WHERE ActConsID = "' + ConsumerID + '"');
     ExecSQL;
    end;

   {build table of technologies using join of necessary tables}
   with query2 do
    begin
     close;
     SQL.clear;
     SQL.add('SELECT DISTINCT * ' +
           'FROM TECHNOLO,PROVIDER,TECHINFO ' +
           'WHERE (TECHNOLO.ProviderID = PROVIDER.ProviderID)' +
           'AND (TECHNOLO.TechID = TECHINFO.TechID)' +
           'AND (TECHNOLO.ProviderID = TECHINFO.ProviderID)' +
           'AND (TECHNOLO.ProviderID = "'+ ProviderID + '")' +
           'AND (TECHNOLO.TechID = "'+ TechID + '")');
     open;
```

```
if RecordCount = 0 then  {Consumer selected an invalid combination
                          of ProviderID and TechID from Menu}
 begin
  send('<center><H1>Cannot Locate Technology</H1>');
  send('<H2>The technology with ProviderID <I>' + ProviderID + '</I>');
  send('and TechID <I>' + TechID + '</I> does not exist.  Please ');
  send('return to the Consumer Menu, and use the "List Technologies" ');
  send('option to find the correct technology.</H2></center>');
 end

else
 begin
 send('<center><H1>DecisionNet Technology Details</h1></center>');
 send('This technology is owned and maintained by its ');
 send('individual provider.  DecisionNet"s purpose is to ');
 send('facilitate access to this technology.');

 if FieldByName('ExcInd').AsString = 'Independent' then
 begin
 send(' When you execute this technology, it will open into a new ');
 send('browser window.  It is important to close that window and ');
 send('return to this frame then ');
 send('click on the button below to go back to the Consumer Menu. ');
 send('This will allow DecisionNet ');
 send('to keep track of certain information about you. If you ');
 send('do not do this, you may be forced to log back in to access ');
 send('DecisionNet again. Thank you.');
 end;

 send('<p>');
 send('<center>');

 send('<TABLE BORDER=6 CELLPADDING=6>');

 send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
 send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
 send('<B>Technology Name:</B>');
 send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
 send(FieldByName('TechName').AsString);

 send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
 send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
 send('<B>Provider Name:</B>');
```

```
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send(FieldByName('pName').AsString);

send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<B>Provider ID:</B>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send(FieldByName('ProviderID').AsString);

send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<B>Technology ID:</B>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send(FieldByName('TechID').AsString);

send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<B>Registered:</B>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send(FieldByName('DateRegistered').AsString);

send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<B>Updated:</B>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send(FieldByName('LastUpdate').AsString);

send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<B>Object Type</B>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send(FieldByName('tObjectType').AsString);

send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<B>Problem Area</B>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send(FieldByName('tProblemArea').AsString);

send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<B>Functional Area</B>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
```

```
send(FieldByName('tFunctionalArea').AsString);

send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<B>Solution Method</B>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send(FieldByName('tSolutionMethod').AsString);

send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<B>Industry Type</B>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send(FieldByName('tIndType').AsString);

send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<B>Organization Type</B>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send(FieldByName('tOrgType').AsString);
send('</TABLE>');
send('</center>');

send('<P>');
send('<B>Purpose:</B> ');
CGIDB1.SendMemo(FieldByName('Purpose'));
send('<P>');

send('<P>');
send('<B>Comments:</B> ');
CGIDB1.SendMemo(FieldByName('Comments'));
send('<P>');
send('<CENTER>');

if FieldByName('ExcInd').AsString = 'Independent' then
 begin
  send('<FORM method=post action="'+
     'http://131.120.39.63/cgi-win/dnet/consumer/execind.exe" ');
  send('TARGET="execind">');
 end
else
 begin
  send('<FORM method=post action="'+
     'http://131.120.39.63/cgi-win/dnet/consumer/null.exe">');
```

```
        end;

        send('<input type="hidden" name="ConsumerID" '+
             'value="'+ConsumerID+'">');
        send('<input type="hidden" name="ProviderID" '+
             'value="'+ProviderID+'">');
        send('<input type="hidden" name="TechID" '+
             'value="'+TechID+'">');
        send('<input type="submit" value="Execute Technology">');
        send('</form>');
        send('</CENTER>');
       end;
       Query2.close;
       end;

    send('<p>');
    sendHR;
    send('<p>');

    {capture ConsumerID, send user to Consumer Menu}
    send('<CENTER>');
    send('<FORM method=post action="'+
     'http://131.120.39.63/cgi-win/dnet/consumer/menu.exe">');
    send('<input type="hidden" name="ConsumerID" '+
     'value="'+ConsumerID+'">');
    send('<input type="submit" value="Registered Consumer Menu">');
    send('</form>');
    send('</CENTER>');

    send('<p>');
    sendHR;

    send('<p><FONT SIZE=-1><I>');
    send( 'This application was created by Steve Earley ');
    send( 'for Professor Hemant Bhargava.<br>' );
    send( 'Generated on ' + webdate(now) );
    send( '</I></FONT></BODY></HTML>' );

    closeStdout;
    closeApp( application );
    end;
end;
end.
```

6. Execute Independent Technologies

```
unit Execind1;    {Consumer Execute Independent Technology script;
                   created by Steve Earley. Last update: 11 Jun 96.}

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, Cgi, DB, DBTables;

type
  TForm1 = class(TForm)
    CGIEnvData1: TCGIEnvData;
    DataSource1: TDataSource;
    Table2: TTable;
    DataSource2: TDataSource;
    Table1: TTable;
    DataSource3: TDataSource;
    Query1: TQuery;
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);

var
    ConsumerID: string;
    ProviderID: string;
    TechID: string;

begin
     with CGIEnvData1 do
```

```
    begin
      websiteINIFilename := paramstr(1);
      application.onException := cgiErrorHandler;
      application.processMessages;

      {Receive hidden fields from about.exe}
      ConsumerID := GetSmallField('ConsumerID');
      ProviderID := GetSmallField('ProviderID');
      TechID := GetSmallField('TechID');

      {update Active Consumer table}
      with Query1 do
       begin
        close;
        SQL.clear;
        sql.add('UPDATE ACT_CONS ');
        sql.add('SET LastActionTime = ''' +DateTimeToStr(Now)+ '''');
        sql.add('WHERE ActConsID = ''' + ConsumerID + '''');
        ExecSQL;
       end;

      {Locate appropriate record in Technology table}
      Table1.open;
      Table1.FindKey([ProviderID,TechID]);

      {Add record to Used Technology table}
      Table2.open;
      Table2.AppendRecord([ProviderID,TechID,ConsumerID,DateTimeToStr(Now),
                DateTimeToStr(Now)]);
      Table2.close;

      {send user to independent technology}
      createStdout;
      bounceToLocation(Table1.FieldByName('tURL').AsString);

      Table1.close;
      closeApp( application );

    end;
end;
end.
```

7. Indexed Search

```
unit Indexed1; {Technology Indexed Search; created by Steve Earley.
        Last updated 2 Jul 96.}

interface

uses
 SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
 Forms, Dialogs, DB, Cgidb, Cgi, DBTables;

type
 TForm1 = class(TForm)
  CGIEnvData1: TCGIEnvData;
  DataSource1: TDataSource;
  Query1: TQuery;
  DataSource2: TDataSource;
  Query2: TQuery;
  DataSource3: TDataSource;
  Query3: TQuery;
  BatchMove1: TBatchMove;
  Table1: TTable;
  DataSource4: TDataSource;
  procedure FormCreate(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);

var
  ConsumerID : string;   {consumer using this program}
  CatField : array[0..5] of string; {fields chosen by user}
```

```
  Category : array[0..5] of string; {categories available for user to
                                     choose from}
  CatUsed : boolean; {flag to show if any categories have been used in query}
  i : smallint; {counter to keep track of array elements}

begin
 with CGIEnvData1 do
 begin
  websiteINIFilename := paramstr(1);
  application.onException := cgiErrorHandler;
  application.processMessages;
  createStdout;
  sendPrologue;

  {Get fields from Consumer Menu}
  ConsumerID := GetSmallField('ConsumerID');
  CatField[0] := GetSmallField('tObjectType');
  CatField[1] := GetSmallField('tProblemArea');
  CatField[2] := GetSmallField('tFunctionalArea');
  CatField[3] := GetSmallField('tSolutionMethod');
  CatField[4] := GetSmallField('tIndType');
  CatField[5] := GetSmallField('tOrgType');

  {Initialize array and flag}
  Category[0] := 'tObjectType';
  Category[1] := 'tProblemArea';
  Category[2] := 'tFunctionalArea';
  Category[3] := 'tSolutionMethod';
  Category[4] := 'tIndType';
  Category[5] := 'tOrgType';
  CatUsed := false;

  {standard header information}
  send( '<HTML><HEAD>' );
  sendTitle( 'DecisionNet Indexed Search' );
  send( '</HEAD><BODY BGCOLOR="80B7B0">' );
  send('<center><H1>DecisionNet Indexed Search</h1></center>');

  {update Active Consumer table}
  with Query1 do
   begin
    close;
    SQL.clear;
```

```
  sql.add('UPDATE ACT_CONS ');
  sql.add('SET LastActionTime = ''' +DateTimeToStr(Now)+ '''');
  sql.add('WHERE ActConsID = ''' + ConsumerID + '''');
  ExecSQL;
 end;

{Build table of technologies using queries on Technolo.db. Query2 checks
 for technologies that meet the user's chosen criteria. Query3 checks for
 technologies that apply to several categories (labeled as "ALL" in the
 database).}
Query2.close;
Query3.close;
Query2.SQL.clear;
Query3.SQL.clear;
Query2.SQL.add('SELECT * FROM TECHNOLO '); {Chooses all technologies
                                            by default}

for i := 0 to 5 do
 begin
  if CatField[i] <> 'ALL' then   {user has selected a category}
   begin
    if CatUsed = false then     {first WHERE clause for SQL queries}
     begin
      Query2.SQL.add('WHERE ('+ Category[i] + ' = ''' + CatField[i] + ''') ');
      Query3.SQL.add('SELECT * FROM TECHNOLO ');
      Query3.SQL.add('WHERE ('+ Category[i] + ' = "ALL") ');
      CatUsed := true;
     end
    else    {subsequent clauses for queries}
     begin
      Query2.SQL.add('AND ('+ Category[i] + ' = ''' + CatField[i] + ''') ');
      Query3.SQL.add('AND ('+ Category[i] + ' = "ALL") ');
     end;
   end;
 end;

 Query2.SQL.add('ORDER BY TechName');
 Query2.open;
 BatchMove1.Execute;   {creates temp1.db; copies query2 to temp1.db}
 Table1.open;          {opens Temp1.db}
 Query2.close;
```

```
if CatUsed = true then    {User chose one or more criteria}
 begin
  with Query3 do
   begin
    SQL.add('ORDER BY TechName');
    open;
    first;
    while not EOF do    {Add records from Query3 to temp1.db}
     begin
      Table1.AppendRecord([Query3.Fields[0],Query3.Fields[1],Query3.Fields[2],
                  Query3.Fields[3],Query3.Fields[4],Query3.Fields[5],
                  Query3.Fields[6],Query3.Fields[7],Query3.Fields[8],
                  Query3.Fields[9],Query3.Fields[10]]);
      next;
     end;
    close;
   end;
end;

if Table1.RecordCount = 0 then    {empty query}
 begin
  send('<center><p><h2>');
  send('Your query did not return a result.  Please try a broader ');
  send('search, with more of the drop-down lists set to "ALL."');
  send('</h2></center>');
 end
else    {Query returned a result}
 begin
  {Print disclaimer}
  send('These technologies are owned and maintained by their ');
  send('individual providers.  DecisionNet"s purpose is to ');
  send('facilitate access to these technologies. The listing below is ');
  send('based on technologies that meet your specific query, along with ');
  send('any technologies that apply to all categories (shown as "ALL").');
  send('<p>');

  {Set-up table headings}
  send('<center>');
  send('<TABLE BORDER=6 CELLPADDING=6>');
  send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');

  send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
  send('<B>Technology Name</B></TD>');
```

```
{Print table headers if categories are queried}
if CatField[0] <> 'ALL' then
send('<TD ALIGN="CENTER"><B>Object Type</B></TD>');

if CatField[1] <> 'ALL' then
send('<TD ALIGN="CENTER"><B>Problem Area</B></TD>');

if CatField[2] <> 'ALL' then
send('<TD ALIGN="CENTER"><B>Functional Area</B></TD>');

if CatField[3] <> 'ALL' then
send('<TD ALIGN="CENTER"><B>Solution Method</B></TD>');

if CatField[4] <> 'ALL' then
send('<TD ALIGN="CENTER"><B>Industry Type</B></TD>');

if CatField[5] <> 'ALL' then
send('<TD ALIGN="CENTER"><B>Organization Type</B></TD>');

with Table1 do
 begin
  first;
  while not EOF do {fill in values for dynamic page using contents of
                    temp1.db}
   begin
    send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
    send('<FORM METHOD = POST  ACTION= ');
    send('"http://131.120.39.63/cgi-win/dnet/consumer/about.exe">');
send('<INPUT TYPE=HIDDEN NAME="ConsumerID" VALUE="'+ConsumerID+'">');
    send('<INPUT TYPE=HIDDEN NAME="ProviderID" ');
    send('VALUE="'+ FieldByName('ProviderID').AsString +'">');
    send('<INPUT TYPE=HIDDEN NAME="TechID" ');
    send('VALUE="'+ FieldByName('TechID').AsString +'">');
    send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
    send('<INPUT TYPE=SUBMIT ');
    send('VALUE="'+ FieldbyName('TechName').AsString +'">');
    send('</TD></FORM>');

    for i := 0 to 5 do
       begin
        if CatField[i] <> 'ALL' then
         begin
          send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
```

```
            send(FieldByName(Category[i]).AsString);
            send('</TD>');
           end;
          end;

        next; {puts cursor on next record}
       end; {end of file for table}
      close;
      Active := False;        {Mark temp1.db for deletion, and delete it}
      DatabaseName := 'DNET';
      TableName := 'TEMP1';
      TableType := ttDefault;
      DeleteTable;
     end;     {Table1}

    send('</TABLE>');
    send('</center>');
   end;    {display of query result table}

  sendHR;
    .
  {capture UserID, send user to Consumer Menu}
  send('<CENTER>');
  send('<FORM method=post action="'+
   'http://131.120.39.63/cgi-win/dnet/consumer/menu.exe">');
  send('<input type="hidden" name="ConsumerID" '+
   'value="'+ConsumerID+'"></td>');
  send('<input type="submit" value="Registered Consumer Menu">');
  send('</form>');
  send('</CENTER>');

  {standard HTML footer information}
  sendHR;
  send('<FONT SIZE=-1><I>');
  send( 'This application was created by Steve Earley');
  send( 'for Professor Hemant Bhargava.<br>' );
  send( 'Generated on ' + webdate(now) );
  send( '</I></FONT></BODY></HTML>' );
  closeStdout;
  closeApp( application );
 end;
end;
end.
```

8. Modify Consumer Information

```
unit Modify1;   {Modify Consumer Information; created by Steve Earley,
                 last updated 2 Jul 96.}

interface

uses
 SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
 Forms, Dialogs, Cgi, DB, DBTables;

type
 TForm1 = class(TForm)
  CGIEnvData1: TCGIEnvData;
  DataSource1: TDataSource;
  Query1: TQuery;
  DataSource2: TDataSource;
  Table1: TTable;
  procedure FormCreate(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);

var
 ConsumerID : string;

begin
 with CGIEnvData1 do
 begin
   {required when this program runs under WebSite}
   webSiteINIFilename := paramstr(1);
   application.onException := cgiErrorHandler;
```

```
application.processMessages;
createStdout;
sendPrologue;

{HTML page header info}
send( '<HTML><HEAD>' );
sendTitle( 'Modify Consumer Information' );
send( '</HEAD><BODY BGCOLOR="80B7B0">' );

{receive input field from Consumer Menu; if user tries to go directly to
program, or tries to modify record as a guest, raise an error}
ConsumerID := getSmallField( 'ConsumerID' );

 if (ConsumerID = CGINotFound) or (ConsumerID = 'guest') then
  begin
   send('<center><H1>Unable to Modify Record</h1>');
   send('<h2>You cannot modify your information unless you have logged ');
   send('in as a valid Consumer other than <I>'+ConsumerID+'</I>. ');
   send('Please <a href="http://131.120.39.66/start.htm">login</a> ');
   send('to continue.</h2></center>');
  end

 else
  begin
   {update Active Consumer table}
    with Query1 do
       begin
          close;
          SQL.clear;
          sql.add('UPDATE ACT_CONS ');
          sql.add('SET LastActionTime = ''' +DateTimeToStr(Now)+ '''');
          sql.add('WHERE ActConsID = ''' + ConsumerID + '''');
          ExecSQL;
       end;

with Table1 do    {puts cursor on correct record in Consumer table}
    begin
       open;
       SetKey;
       FieldByName('ConsumerID').AsString := ConsumerID;
       GotoKey;
    end;
```

{Modification form}

```
send('<CENTER><h1>Modify Consumer Information</h1></CENTER> ');

send('<B>Directions:</B>');

send('<OL><LI>Please enter your current password in the "Old Password" ');
send('field (even if you are not changing your password).');

send('<LI>For the remaining fields, fill in any information that ');
send('you want to change, then press the "Modify Record" button. If you ');
send('do not want to change the information as shown, please do not ');
send('delete the entries in these fields.');

send('<LI>You cannot change your Consumer ID using this page.  If you ');
send('need to change your Consumer ID, please feel free to contact us.');

send('<LI>If you change your password, ');
send('you must type in all three password fields (old, new, and new ');
send('again) for the change to take effect.</OL>');

send('<center> ');
send('<FORM method=post action= ');
  send('"http://131.120.39.63/cgi-win/dnet/consumer/modifya.exe"> ');
send('<input type="hidden" name="ConsumerID" value="' +ConsumerID+ '">');
send('<TABLE BORDER=6 CELLPADDING=6> ');
send('<td>Old Password:</td><td><input type="password" ');
  send('name="cPassword" size=10></td> ');
send('<TR><TD>New Password:</TD><TD><input type="password" ');
  send('name="cPassword2"size=10> </td> ');
send('<TR><TD>Re-type New Password:</TD><TD><input type="password" ');
  send('name="cPassword3"  size=10> </td> ');
send('<TR><TD> Last name:</TD> <td> <input type="text" ');
  send('name="cLastName" size=30 value=');
  send('"" +Table1.FieldByName('cLastName').AsString+ '"></td> ');
send('<TR><TD>  First name: </TD> <td><input type="text" ');
  send('name="cFirstName" size=30 value=');
  send('"" +Table1.FieldByName('cFirstName').AsString+ '"></td> ');
send('<TR><TD>  Email Address: </TD> <td><input type="text" ');
  send('name="cEMailAddress" size=50 value=');
  send('"" +Table1.FieldByName('cEMailAddress').AsString+ '"></td> ');
send('</TABLE> ');

send('<P>');
```

```
    send('<input type="submit" value= "Modify Record"> ');
    send('<input type="reset" value="Clear Form">');
    send('</form></center>');

    {If user cancels, capture ConsumerID, send user to Consumer Menu}
    send('<center><FORM method=post action="'+
      'http://131.120.39.63/cgi-win/dnet/consumer/menu.exe">');
    send('<input type="hidden" name="ConsumerID" '+
      'value="'+ConsumerID+'"></td>');
    send('<input type="submit" value="Cancel, Return to Consumer Menu">');
    send('</form>');
    send('</CENTER>');
     end;

    {HTML page footer info}
    send('<HR><I><FONT SIZE=-1>');
    send( 'This application was created by Steve Earley for Professor ');
    send( 'Hemant Bhargava.<BR>' );
    send( 'Generated on ' + webdate(now) );
    send( '</I></FONT></BODY></HTML>' );
    closeStdout;

  end;
end;
end.
```

9. Withdraw from DecisionNet

```
unit Withdra1; {Consumer Withdraw; created by Steve Earley; updated: 7 Jun 96.}

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, Cgi, DBTables, DB;

type
  TForm1 = class(TForm)
    DataSource1: TDataSource;
    Table1: TTable;
    CGIEnvData1: TCGIEnvData;
    DataSource2: TDataSource;
    Query1: TQuery;
    DataSource3: TDataSource;
    Query2: TQuery;
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);

var
  ConsumerID : string;

begin
  with CGIEnvData1 do
  begin
    { required when this program runs under WebSite}
    webSiteINIFilename := paramstr(1);
```

```
 application.onException := cgiErrorHandler;
 application.processMessages;

 {Get ConsumerID field from menu.exe}
 ConsumerID := getSmallField( 'ConsumerID' );

 {standard dynamic HTML header information}
 createStdout;
 sendPrologue;
 send( '<HTML><HEAD>' );
 sendTitle( 'Withdraw as DecisionNet Consumer' );
 send( '</HEAD><BODY BGCOLOR="80B7B0">' );

if ConsumerID = 'guest' then    {disallow changes to guest record}
 begin
 send('<CENTER><h1>Cannot Remove Record</h1>');
 send('<h2>You do not have permission to modify the <I>guest</I> ');
 send('account.  If you would like to enter your own information, ');
 send('please go to our <A HREF="http://dnet.sm.nps.navy.mil/start.htm">');
 send('Start DecisionNet Page</A>.</h2></CENTER><HR>');

 {If user cancels, capture ConsumerID, send user to Consumer Menu}
 send('<center><FORM method=post action="'+
   'http://131.120.39.63/cgi-win/dnet/consumer/menu.exe">');
 send('<input type="hidden" name="ConsumerID" '+
   'value="'+ConsumerID+'"></td>');
 send('<input type="submit" value="Cancel, Return to Consumer Menu">');
 send('</form>');
 send('</CENTER>');
 end

else    {ok to delete record}
 begin

 with Table1 do    {search for user in Active Consumer table;
                    GotoKey returns True if already logged in}
   begin
     open;
     SetKey;
     FieldByName('ActConsID').AsString := ConsumerID;
     GoToKey;
   end;
```

```
if Table1.GoToKey = True then
  begin
    {Delete user from Active Consumer table}
    with Query1 do
      begin
       close;
       SQL.clear;
       sql.add('DELETE FROM ACT_CONS ');
       sql.add('WHERE ActConsID = ''' + ConsumerID + '''');
       ExecSQL;
      end;
  end;

    {Delete user from Consumer table}
    with Query2 do
      begin
       close;
       SQL.clear;
       sql.add('DELETE FROM CONSUMER ');
       sql.add('WHERE ConsumerID = ''' + ConsumerID + '''');
       ExecSQL;
      end;

Table1.close;

send('<center><h1>Withdraw as DecisionNet Consumer</h1></center>');
send('<HR><P>You are now removed from the system. ');
send(' We are sorry to see you go. Please feel free to register ');
send('with us again in the future. ');
send(' We hope to have a fully-functional system by the end of June, ');
send('and we invite your comments for any improvements or additions ');
send('you would like to see. ');
send(' Thank you for using DecisionNet.');

send('<H3>For DSS or overall system questions, please contact:</H3>');
send('<P><CENTER><A HREF="mailto:bhargava@nps.navy.mil">');
send('Prof. Hemant Bhargava: <I>bhargava@nps.navy.mil</I></A></CENTER>');

send('<H3>For technical questions and support, please contact:</H3>');
send('<P><CENTER><A HREF="mailto:shearley@nps.navy.mil">');
send('Steve Earley: <I>shearley@nps.navy.mil</A></CENTER>');

end;
```

```
    {standard dynamic HTML footer information}
    send('<p>');
    sendHR;
    send('<p><i><font size=-1>');
    send( 'This application was created by Steve Earley ');
    send( 'for Professor Hemant Bhargava.<br>' );
    send( 'Generated on ' + webdate(now) );
    send( '</font></i></BODY></HTML>' );
    closeStdout;

  end;
end;
end.
```

10. Logout Script

```
unit Logout1; {Consumer Logout; created by Steve Earley; updated: 14 May 96.}

interface

uses
 SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
 Forms, Dialogs, Cgi, DBTables, DB;

type
 TForm1 = class(TForm)
  DataSource1: TDataSource;
  Table1: TTable;
  CGIEnvData1: TCGIEnvData;
  DataSource2: TDataSource;
  Query1: TQuery;
  procedure FormCreate(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);

var
 ConsumerID : string;

begin
 with CGIEnvData1 do
 begin
  { required when this program runs under WebSite}
  webSiteINIFilename := paramstr(1);
  application.onException := cgiErrorHandler;
  application.processMessages;
```

```
{Get ConsumerID field from menu.exe}
ConsumerID := getSmallField( 'ConsumerID' );

{standard dynamic HTML header information}
createStdout;
sendPrologue;
send( '<HTML><HEAD>' );
sendTitle( 'DecisionNet Consumer Logout' );
send( '</HEAD><BODY BGCOLOR="80B7B0">' );

with Table1 do   {search for user in Active Consumer table;
                  GotoKey returns True if already logged in}
  begin
    open;
    SetKey;
    FieldByName('ActConsID').AsString := ConsumerID;
    GoToKey;
  end;

if Table1.GoToKey = True then
  begin
    {Delete user from Active Consumer table}
    with Query1 do
      begin
       close;
       SQL.clear;
       sql.add('DELETE FROM ACT_CONS ');
       sql.add('WHERE ActConsID = ''' + ConsumerID + '''');
       ExecSQL;
      end;
  end;

Table1.close;

send('<center><h1>DecisionNet Logout</h1></center>');
send('<HR><P>You are now logged out of the system. ');
send(' We hope to have a fully-functional system by the end of June, ');
send('and we invite your comments for any improvements or additions ');
send('you would like to see. ');
send(' Thank you for using DecisionNet.');

send('<H3>For DSS or overall system questions, please contact:</H3>');
send('<P><CENTER><A HREF="mailto:bhargava@nps.navy.mil">');
```

```
    send('Prof. Hemant Bhargava: <I>bhargava@nps.navy.mil</I></A></CENTER>');

    send('<H3>For technical questions and support, please contact:</H3>');
    send('<P><CENTER><A HREF="mailto:shearley@nps.navy.mil">');
    send('Steve Earley: <I>shearley@nps.navy.mil</A></CENTER>');

    {standard dynamic HTML footer information}
    send('<p>');
    sendHR;
    send('<p><i><font size=-1>');
    send( 'This application was created by Steve Earley ');
    send( 'for Professor Hemant Bhargava.<br>' );
    send( 'Generated on ' + webdate(now) );
    send( '</I></font></BODY></HTML>' );
    closeStdout;

  end;
end;
end.
```

C. PROVIDER SCRIPTS

1. Provider Registration

```
unit Registe1; {Provider Registration; created by Steve Earley;
               updated 10 Jun 96.}

interface

uses
 SysUtils, WinTypes, WinProcs, Messages, Classes,
 Forms, Cgi, DB, DBTables;

type
 TForm1 = class(TForm)
  DataSource1: TDataSource;
  Table1: TTable;
  CGIEnvData1: TCGIEnvData;
  DataSource2: TDataSource;
  Table2: TTable;
  DataSource3: TDataSource;
  Table3: TTable;
  procedure FormCreate(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);

var
 ProviderID : string;
 pPassword : string;
 pPassword2 : string;
```

```
  pName : string;
  pURL : string;
  pEMailAddress : string;

begin
  with CGIEnvData1 do
  begin
    { required when this program runs under WebSite}
    webSiteINIFilename := paramstr(1);
    application.onException := cgiErrorHandler;
    application.processMessages;

    {receive input fields from HTML form}
    ProviderID := getSmallField( 'ProviderID' );
    pPassword := getSmallField( 'pPassword' );
    pPassword2 := getSmallField( 'pPassword2' );
    pName := getSmallField( 'pName' );
    pURL := getSmallField( 'pURL' );
    pEMailAddress := getSmallField( 'pEMailAddress' );

    {HTML page header info}
    createStdout;
    sendPrologue;

    send( '<HTML><HEAD>' );
    sendTitle( 'DecisionNet Provider Registration' );
    send( '</HEAD><BODY BGCOLOR="80B7B0">' );

    with Table1 do   {puts cursor on correct record in Provider table}
      begin
        open;
        SetKey;
        FieldByName('ProviderID').AsString := ProviderID;
        GotoKey;
      end;

    with Table2 do   {puts cursor on correct record in Provider Mirror table}
      begin
        open;
        SetKey;
        FieldByName('ProviderID').AsString := ProviderID;
        GotoKey;
      end;
```

```
if (Table1.GoToKey = True) or (Table2.GoToKey = True)
  or (ProviderID = CGINotFound) then

  {User's ID choice already exists on Provider or ProvMirror table,
  or no ProviderID entered on form}
  begin
    send('<center><h1>Invalid Provider ID</h1>');
    send('<h2>Sorry, but you cannot use the Provider ID you have ');
    send('chosen.  Please try another one.</h2></center>');
    send('<p>');
    sendHR;
    send('<p>');

    send('<CENTER> ' +
     '<TABLE BORDER=6 CELLPADDING=6> ' +
     '<TR ALIGN="CENTER" VALIGN=MIDDLE> ' +
           '<TD ALIGN="CENTER" VALIGN=MIDDLE> ' +
           '<A HREF="http://dnet.sm.nps.navy.mil/provider/register.htm"> ' +
     'Return to Provider Registration</A></TD>');
    send('</TR></TABLE></CENTER>');
    send('<p>');
  end

else {valid Provider ID chosen}
  begin
    if pPassword <> pPassword2 then {bad registration}
     begin
      send('<center><h1>Password Mismatch</h1>');
      send('<h2>Please verify your password choice, and try ');
      send('again.</h2></center>');
      send('<p>');
      sendHR;
      send('<p>');

      send('<CENTER> ' +
        '<TABLE BORDER=6 CELLPADDING=6> ' +
        '<TR ALIGN="CENTER" VALIGN=MIDDLE> ' +
              '<TD ALIGN="CENTER" VALIGN=MIDDLE> ' +
              '<A HREF="http://dnet.sm.nps.navy.mil/Provider/register.htm">' +
     'Return to Provider Registration</A></TD>');
      send('</TR></TABLE></CENTER>');
      send('<p>');
     end
```

```
    else {good registration}
      begin
        {input record into Provider table}
        Table1.AppendRecord([ProviderID, pPassword, pName,
              pURL, pEMailAddress]);
        Table1.close;

        {input record into Provider Mirror table}
        Table2.AppendRecord([ProviderID, pPassword, pName,
              pURL, pEMailAddress]);
        Table2.close;

        {input record into Active Provider table}
        Table3.open;
        Table3.AppendRecord([ProviderID, DateTimeToStr(Now),
              DateTimeToStr(Now)]);
        Table3.close;

        send('<center><h1>Welcome to DecisionNet!</h1>');
        send('<h2>You are now registered under the ' +
          'User Name <I>' + ProviderID  + '</I>. ');
        send(' Thank you for using DecisionNet.</h2></center>');
        send('<p>');
        sendHR;
        send('<p>');

        {capture ProviderID, send user to Provider Menu}
        send('<CENTER>');
        send('<FORM method=post action="'+
          'http://131.120.39.63/cgi-win/dnet/provider/menu.exe">');
        send('<input type="hidden" name="ProviderID" '+
          'value="'+ProviderID+'"></td>');
        send('<input type="submit" value="Registered Provider Menu">');
        send('</form>');
        send('</CENTER>');
        send('<p>');
      end;
  end;

{HTML page footer info}
send('<HR>');
send( 'This application was created by Steve Earley for Professor ');
send( 'Hemant Bhargava.<br>' );
```

```
    send( 'Generated on ' + webdate(now) );
    send( '</BODY></HTML>' );
    closeStdout;
  end;
end;
end.
```

2. Provider Login

```
unit Login1; {Provider Login; created by Steve Earley; updated: 20 May 96.}

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, Cgi, DBTables, DB;

type
  TForm1 = class(TForm)
    DataSource1: TDataSource;
    DataSource2: TDataSource;
    Table1: TTable;
    Table2: TTable;
    CGIEnvData1: TCGIEnvData;
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);

var
  ProviderID : string;
  pPassword : string;

begin
  with CGIEnvData1 do
  begin
    { required when this program runs under WebSite}
    webSiteINIFilename := paramstr(1);
    application.onException := cgiErrorHandler;
```

```
application.processMessages;

{input fields from Login form (start.htm)}
ProviderID := getSmallField( 'ProviderID' );
pPassword := getSmallField( 'pPassword' );

{standard dynamic HTML header information}
createStdout;
sendPrologue;
send( '<HTML><HEAD>' );
sendTitle( 'DecisionNet Provider Login' );
send( '</HEAD><BODY BGCOLOR="80B7B0">' );

with Table1 do   {puts cursor on correct record in Provider table;
                  GotoKey returns True if record is valid}
  begin
    open;
    SetKey;
    FieldByName('ProviderID').AsString := ProviderID;
    GotoKey;
  end;

if (Table1.FieldByName('pPassword').AsString <> pPassword) or
 (Table1.GoToKey = False) then   {password does not match ProviderID, or
                                  user not in Provider table}

  begin
    send('<center><h1>Incorrect Login!</h1></center>');
    send('<h2>Please verify that your User Name and Password ' +
      'are correct, then try again.</h2>');
    send('<p>');
    sendHR;
    send('<p>');

    send('<CENTER> ' +
      '<TABLE BORDER=6 CELLPADDING=6> ' +
      '<TR ALIGN="CENTER" VALIGN=MIDDLE> ' +
        '<TD ALIGN="CENTER" VALIGN=MIDDLE> ' +
        '<A HREF="http://dnet.sm.nps.navy.mil/start.htm"> ' +
      'DecisionNet Start Page</A></TD>');
    send('</TR></TABLE></CENTER>');
  end
```

```
else  {ProviderID matches pPassword}
  begin
     with Table2 do   {search for user in Active Provider table;
                GotoKey returns True if already logged in}
       begin
         open;
         SetKey;
         FieldByName('ActProvID').AsString := ProviderID;
         GoToKey;
       end;

       if Table2.GoToKey = True then   {already logged in}
         begin
           send('<center><h1>Already logged in</h1>');
           send('<h2>You were previously logged in under the ');
           send('User Name <I>' + ProviderID  + '</I> ');
           send('and did not logout. Your last action was at ');
           send(Table2.FieldByName('LastActionTime').AsString +'.');
           send(' There is no need to login again.');
           send('<P>');
           send(' Thank you for using DecisionNet.</h2></center>');
           send('<p>');
         end
       else
         begin
           {put user in Active Provider table}
           with Table2 do
           begin
           open;
           AppendRecord([ProviderID, DateTimeToStr(Now),
             DateTimeToStr(Now)]);
           end;

           send('<center><h1>Welcome to DecisionNet!</h1>');
           send('<h2>You are logged in under the ' +
             'User Name <I>' + ProviderID  + '</I>. ');
           send(' Thank you for using DecisionNet.</h2></center>');
           send('<p>');
         end;

    {capture ProviderID, send user to Provider Menu}
    send('<CENTER>');
    send('<FORM method=post action="'+
```

```
            'http://131.120.39.63/cgi-win/dnet/provider/menu.exe">');
        send('<input type="hidden" name="ProviderID" '+
            'value="'+ProviderID+'"></td>');
        send('<input type="submit" value="Registered Provider Menu">');
        send('</form>');
        send('</CENTER>');
      end;

    Table1.close;
    Table2.close;

    {standard dynamic HTML footer information}
    send('<p>');
    sendHR;
    send('<p><I><FONT SIZE=-1>');
    send( 'This application was created by Steve Earley ');
    send( 'for Professor Hemant Bhargava.<br>' );
    send( 'Generated on ' + webdate(now) );
    send( '</I></FONT></BODY></HTML>' );
    closeStdout;

  end;
end;
end.
```

3. Provider Menu

```
unit Menu1;   {Active Provider menu; created by Steve Earley,
          Last updated 20 Jun 96}

interface

uses

 SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
 Forms, Dialogs, DB, Cgidb, Cgi, DBTables;

type
 TForm1 = class(TForm)
  CGIEnvData1: TCGIEnvData;
  Query1: TQuery;
  DataSource1: TDataSource;
  procedure FormCreate(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);

var
  ProviderID: string;

begin
 with CGIEnvData1 do
  begin
    websiteINIFilename := paramstr(1);
    application.onException := cgiErrorHandler;
    application.processMessages;
```

```
createStdout;
sendPrologue;

{standard header information}
send( '<HTML><HEAD>' );
sendTitle( 'DecisionNet Provider Menu' );
send( '</HEAD><BODY BGCOLOR="80B7B0">' );

{Get ProviderID from start page; if user tries to go directly to
menu, raise an error}
ProviderID := GetSmallField('ProviderID');

if ProviderID = CGINotFound then
 begin
  send('<center><H1>Not Logged In</h1>');
  send('<h2>You are currently not logged in to DecisionNet. ');
  send('Please <a href="http://131.120.39.66/start.htm">login</a> ');
  send('to continue.</h2></center>');
 end

else
 begin

{update Active Provider table}
 with Query1 do
  begin
    close;
    SQL.clear;
    sql.add('UPDATE ACT_PROV ');
    sql.add('SET LastActionTime = ''' +DateTimeToStr(Now)+ '''');
    sql.add('WHERE ActProvID = ''' + ProviderID + '''');
    ExecSQL;
  end;

 send('<center><H1>DecisionNet Provider Menu</h1></center>');
 sendHR;

{Table of Technology-related options for Provider}
 send('<H2>Technology:</H2>');

 send('<CENTER> ');
 send('<TABLE BORDER=6 CELLPADDING=6>');
 send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
```

```
  send('<FORM METHOD = POST  ACTION= ');
  send('"http://131.120.39.63/cgi-win/dnet/provider/regtech.exe">');
  send('<INPUT TYPE=HIDDEN NAME="ProviderID" VALUE="'+ProviderID+'">');
  send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
  send('<INPUT TYPE=SUBMIT VALUE="Register Technology"></TD></FORM>');

  send('<FORM METHOD = POST  ACTION= ');
  send('"http://131.120.39.63/cgi-win/dnet/provider/updttech.exe">');
  send('<INPUT TYPE=HIDDEN NAME="ProviderID" VALUE="'+ProviderID+'">');
  send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
  send('<INPUT TYPE=SUBMIT VALUE="Update Technology"></TD></FORM>');

  send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');

  send('<FORM METHOD = POST  ACTION= ');
  send('"http://131.120.39.63/cgi-win/dnet/provider/techinfo.exe">');
  send('<INPUT TYPE=HIDDEN NAME="ProviderID" VALUE="'+ProviderID+'">');
  send('<TD ALIGN="CENTER" VALIGN=MIDDLE> ');
send('<INPUT TYPE=SUBMIT VALUE="Technology Information"></TD></FORM>');

  send('<FORM METHOD = POST  ACTION= ');
  send('"http://131.120.39.63/cgi-win/dnet/provider/wdtech.exe">');
  send('<INPUT TYPE=HIDDEN NAME="ProviderID" VALUE="'+ProviderID+'">');
  send('<TD ALIGN="CENTER" VALIGN=MIDDLE> ');
 send('<INPUT TYPE=SUBMIT VALUE="Withdraw Technology"></TD></FORM>');

  send('</TR></TABLE></CENTER>');
  send('<P>');

  send('<FONT SIZE=-1>');
  send('<UL>');
  send('<LI><DT><STRONG>Register Technology</STRONG>');
  send('<DD>Register a new decision support technology with DecisionNet');
  send('<LI><DT><STRONG>Update Technology</STRONG>');
  send('<DD>Update a decision technology being provided on DecisionNet');
  send('<LI><DT><STRONG>Technology Information</STRONG>');
  send('<DD>Get data on the customer usage of provided technologies');
  send('<LI><DT><STRONG>Withdraw Technology</STRONG>');
  send('<DD>Withdraw a decision technology from DecisionNet');
  send('</UL>');
  send('</FONT>');

  sendHR;
```

{Table of Provider account-related options for Provider}

```
send('<H2>Provider Account:</H2>');

send('<CENTER> ');
send('<TABLE BORDER=6 CELLPADDING=6>');
send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');

send('<FORM METHOD = POST  ACTION= ');
send('"http://131.120.39.63/cgi-win/dnet/provider/modify.exe">');
send('<INPUT TYPE=HIDDEN NAME="ProviderID" VALUE="'+ProviderID+'">');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<INPUT TYPE=SUBMIT VALUE="Modify Information"></TD></FORM>');

send('<FORM METHOD = POST  ACTION= ');
send('"http://131.120.39.63/cgi-win/dnet/provider/null.exe">');
send('<INPUT TYPE=HIDDEN NAME="ProviderID" VALUE="'+ProviderID+'">');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<INPUT TYPE=SUBMIT VALUE="Account Information"></TD></FORM>');

send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');

send('<FORM METHOD = POST  ACTION= ');
send('"http://131.120.39.63/cgi-win/dnet/provider/withdraw.exe">');
send('<INPUT TYPE=HIDDEN NAME="ProviderID" VALUE="'+ProviderID+'">');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE> ');
send('<INPUT TYPE=SUBMIT VALUE="Withdraw from DNet"></TD></FORM>');

send('<FORM METHOD = POST  ACTION= ');
send('"http://131.120.39.63/cgi-win/dnet/provider/logout.exe">');
send('<INPUT TYPE=HIDDEN NAME="ProviderID" VALUE="'+ProviderID+'">');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE> ');
send('<INPUT TYPE=SUBMIT VALUE="Logout"></TD></FORM>');

send('</TR></TABLE></CENTER>');
send('<P>');

send('<FONT SIZE=-1>');
send('<UL>');
send('<LI><DT><STRONG>Modify Information</STRONG>');
send('<DD>Change Provider registration data');
send('<LI><DT><STRONG>Account Information</STRONG>');
send('<DD>Get information about your account with DecisionNet');
send('<LI><DT><STRONG>Withdraw from DNet</STRONG>');
```

```
send('<DD>Withdraw a Provider account from DecisionNet');
send('<DD><DD>Note: you must withdraw all registered technologies ');
send('prior to withdrawing your account');
send('<LI><DT><STRONG>Logout</STRONG>');
send('<DD>End your current session');
send('</UL>');
send('</FONT>');

sendHR;

{Other options for Provider}
send('<H2>Other Functions:</H2>');

  {List Technologies form}
send('<H3>List Technologies</H3>');
send('Obtain a listing of all technologies sorted by: ');
send('<FORM METHOD=POST ACTION= ');
send('"http://131.120.39.63/cgi-win/dnet/provider/listtech.exe">');
send('<CENTER>');
send(' <SELECT NAME="sortkey" TYPE = "text" SIZE=1>');
  send('<OPTION SELECTED VALUE="TechName">Technology Name');
  send('<OPTION VALUE="pName">Provider Name');
  send('<OPTION VALUE="tObjectType">Object Type');
  send('<OPTION VALUE="tProblemArea">Problem Type');
send('</SELECT><P><P>');
send('<INPUT TYPE=SUBMIT VALUE="List Technologies" ALIGN="MIDDLE">');
send('<INPUT TYPE=HIDDEN NAME="ProviderID" VALUE="'+ProviderID+'">');
send('</FORM>');
send('</CENTER>');
send('<P>');

  {Browse Taxonomy Form}
send('<H3>Browse Taxonomy</H3>');
send('Explore the hierarchies used in our search algorithm.');
send('<P><P>');
send('<FORM METHOD=POST ACTION= ');
send('"http://131.120.39.63/cgi-win/dnet/provider/browstax.exe">');
send('<CENTER>');
send('<INPUT TYPE=SUBMIT VALUE="Browse Taxonomy" ALIGN=middle>');
send('<INPUT TYPE=HIDDEN NAME="ProviderID" VALUE="'+ProviderID+'">');
send('</FORM>');
send('</CENTER>');
send('<P>');
```

```
        end;
        sendHR;

          {standard footer information}
        send('<p><I><FONT SIZE=-1>');
        send( 'This application was created by Steve Earley ');
        send( 'for Professor Hemant Bhargava.<br>' );
        send( 'Generated on ' + webdate(now) );

        send( '</FONT></I></BODY></HTML>' );

        closeStdout;
        closeApp( application );

      end;
end;
end.
```

4. Register Technology Scripts

```
unit Regtech1; {Technology Registration Form; created by Steve Earley;
        updated 10 Jun 96.}

interface

uses
 SysUtils, WinTypes, WinProcs, Messages, Classes,
 Forms, Cgi, DB, DBTables;

type
 TForm1 = class(TForm)
  DataSource1: TDataSource;
  CGIEnvData1: TCGIEnvData;
  DataSource2: TDataSource;
  Table1: TTable;
  DataSource3: TDataSource;
  Table2: TTable;
  Query1: TQuery;
  procedure FormCreate(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);

var
 ProviderID : string;
 TechID : string;
 TechIDInt : integer;

begin
 with CGIEnvData1 do
```

```
begin
  { required when this program runs under WebSite}
  webSiteINIFilename := paramstr(1);
  application.onException := cgiErrorHandler;
  application.processMessages;

  {Standard Header Information}
  createStdout;
  sendPrologue;

  send( '<HTML><HEAD>' );
  sendTitle( 'DecisionNet Technology Registration Form' );
  send( '</HEAD><BODY BGCOLOR="80B7B0">' );

  {receive input field from Provider Menu; if user tries to go directly to
  program, or tries to register technology as a guest, raise an error}
  ProviderID := getSmallField( 'ProviderID' );

  if (ProviderID = CGINotFound) or (ProviderID = 'guest') then
   begin
    send('<center><H1>Not Logged In</h1>');
    send('<h2>You cannot register a technology unless you have logged ');
    send('in as a valid Provider other than <I>'+ProviderID+'</I>. ');
    send('Please <a href="http://131.120.39.66/start.htm">login</a> ');
    send('to continue.</h2></center>');
   end

  else
   begin

  {update Active Provider table}
  with Query1 do
   begin
    close;
    SQL.clear;
    sql.add('UPDATE ACT_PROV ');
    sql.add('SET LastActionTime = ''' +DateTimeToStr(Now)+ '''');
    sql.add('WHERE ActProvID = ''' + ProviderID + '''');
    ExecSQL;
   end;

  with Table1 do   {Open TechMirror Table, determine which TechID number
                    to use next}
```

```
begin
  TechIDInt := 1;
  open;

  {If Provider is in TechMirror table, Increment TechID until last
   registered Technology is found}
  while FindKey([ProviderID,IntToStr(TechIDInt)]) do
    begin
      TechIDInt := TechIDInt + 1;
    end;
  TechID := IntToStr(TechIDInt);
  close;
end;

send('<center><h1>Technology Registration Form</h1></center>');
send('Please fill in all of the following fields, then press the ');
send('"Register" button. For each category, choose the term that ');
send('best describes your technology. These terms will later be used ');
send('in an indexed search of technologies. Your Technology ID number ');
send('should be up to a three-digit number -- the first technology you ');
send('register will be 1, the second will be 2, and so on. It is best ');
send('to use the number shown in the form to avoid any key violations ');
send('in our database. If you have a technology that was registered and ');
send('subsequently withdrawn from our database, you cannot reuse its ');
send('Technology ID number.');

{Registration Form}
send('<center>');
send('<FORM method=post action=');
send('"http://131.120.39.63/cgi-win/dnet/provider/regteca.exe">');

send('<TABLE BORDER=6 CELLPADDING=6>');

send('<TR><TD>Provider ID:');
send('</TD><td><input type="text" name="ProviderID" ');
send('size=15 maxlength=15 value="' +ProviderID+ '"></td>');

send('<TR><TD>Technology ID:');
send('</TD><td><input type="text" name="TechID" ');
send('size=3 maxlength=3 value="' +TechID+ '"></td>');

send('<TR><TD>Technology Name:');
send('</TD> <td><input type="text" name="TechName" ');
```

```
send('size=20 maxlength=20></td>');

send('<TR><TD>Object Type:');
send('</td><td><SELECT NAME="tObjectType" TYPE="text" SIZE=1>');

with Table2 do    {Check Taxonomy (master.db) for Option Entries}
 begin
  open;
  first; {puts cursor on first record in table}

  while not EOF do
   begin
    if FieldByName('Parent').AsString = 'Object Type' then
     begin
      send('<OPTION VALUE= "'+FieldByName('Child').AsString+'">');
      send(FieldByName('Child').AsString);
     end;
      next; {puts cursor on next record}
   end;
  close;
 end;

send('</SELECT></TD>');

send('<TR><TD>Problem Area:');
send('</TD> <td> <SELECT NAME="tProblemArea" SIZE=1>');
send('<OPTION SELECTED VALUE="ALL">ALL');

with Table2 do    {Check Taxonomy (master.db) for Option Entries}
 begin
  open;
  first; {puts cursor on first record in table}

  while not EOF do
   begin
    if FieldByName('Parent').AsString = 'Problem Area' then
       begin
       send('<OPTION VALUE= "'+FieldByName('Child').AsString+'">');
       send(FieldByName('Child').AsString);
       end;
     next; {puts cursor on next record}
   end;
  close;
```

```
  end;

send('</SELECT></TD>');

send('<TR><TD>Functional Area:');
send('</td><td><SELECT NAME="tFunctionalArea" TYPE="text" SIZE=1>');
send('<OPTION SELECTED VALUE="ALL">ALL');

with Table2 do    {Check Taxonomy (master.db) for Option Entries}
  begin
    open;
    first; {puts cursor on first record in table}

    while not EOF do
      begin
        if FieldByName('Parent').AsString = 'Functional Area' then
           begin
           send('<OPTION VALUE= "'+FieldByName('Child').AsString+'">');
           send(FieldByName('Child').AsString);
           end;
        next; {puts cursor on next record}
      end;
    close;
  end;

send('</SELECT></TD>');

send('<TR><TD>Solution Method:');
send('</TD> <td><SELECT NAME="tSolutionMethod" SIZE=1>');
send('<OPTION SELECTED VALUE="ALL">ALL');

with Table2 do    {Check Taxonomy (master.db) for Option Entries}
  begin
    open;
    first; {puts cursor on first record in table}

    while not EOF do
      begin
        if FieldByName('Parent').AsString = 'Solution Method' then
           begin
            send('<OPTION VALUE= "'+FieldByName('Child').AsString+'">');
            send(FieldByName('Child').AsString);
           end;
```

```
      next; {puts cursor on next record}
     end;
    close;
  end;

send('</SELECT></TD>');

send('<TR><TD>Industry Type:');
send('</TD> <td><SELECT NAME="tIndType" SIZE=1>');
send('<OPTION SELECTED VALUE="ALL">ALL');

with Table2 do    {Check Taxonomy (master.db) for Option Entries}
  begin
    open;
    first; {puts cursor on first record in table}

    while not EOF do
     begin
      if FieldByName('Parent').AsString = 'Industry Type' then
         begin
         send('<OPTION VALUE= "'+FieldByName('Child').AsString+'">');
         send(FieldByName('Child').AsString);
         end;
      next; {puts cursor on next record}
     end;
    close;
  end;

send('</SELECT></TD>');

send('<tr><td>Organization Type:');
send('</td><td><SELECT NAME="tOrgType" SIZE=1>');
send('<OPTION SELECTED VALUE="ALL">ALL');

with Table2 do    {Check Taxonomy (master.db) for Option Entries}
  begin
    open;
    first; {puts cursor on first record in table}

    while not EOF do
     begin
      if FieldByName('Parent').AsString = 'Organization Type' then
         begin
```

```
        send('<OPTION VALUE= "'+FieldByName('Child').AsString+'">');
        send(FieldByName('Child').AsString);
        end;
      next; {puts cursor on next record}
    end;
  close;
end;

send('</SELECT></TD>');

send('<TR><TD>URL:');
send('</TD><td><input type="text" name="tURL" ');
send('size=50 maxlength=255></td>');

send('<TR><TD>Technology Type:');
send('</TD><td><input type="radio" checked name="ExcInd" ');
send('value="Independent"> Independent');
send('<input type="radio" name="ExcInd" value="Exclusive"> Exclusive</td>');

send('<TR><TD>Purpose:</TD>');
send('<TD><textarea rows=5 cols=50 name="Purpose">');
send('Please provide a brief statement of ');
send('your technology"s primary function.</textarea>');

send('<TR><TD>Comments:</TD>');
send('<TD><textarea rows=8 cols=50 name="Comments">');
send('Please provide any details you would like to ');
send('pass on to the user.  Consumers will use this ');
send('information (and the Purpose field above) to ');
send('decide if your technology is right for them. ');
send('These fields will also be used in keyword ');
send('searches for technologies.</textarea>');

send('</TABLE>');
send('<P>');
send('<input type="submit" value= "Register">');
send('<input type="reset" value="Clear Form"></center>');
send('</form>');
send('<P>');

Table1.close;
```

```
    {If user cancels, capture ProviderID, send user to Provider Menu}
    send('<CENTER>');
    send('<FORM method=post action="'+
      'http://131.120.39.63/cgi-win/dnet/provider/menu.exe">');
    send('<input type="hidden" name="ProviderID" '+
      'value="'+ProviderID+'"></td>');
    send('<input type="submit" value="Cancel, Return to Provider Menu">');
    send('</form>');
    send('</CENTER>');
     end;

    {Standard footer information}
    sendHR;
    send('<p><I><font size=-1>');
    send( 'This application was created by Steve Earley for Professor ');
    send( 'Hemant Bhargava.<br>' );
    send( 'Generated on ' + webdate(now) );
    send( '</i></font></BODY></HTML>' );
    closeStdout;

  end;
end;
end.
```

```
unit Regteca1; {Technology Registration; created by Steve Earley;
        updated 7 Jun 96.}

interface

uses
 SysUtils, WinTypes, WinProcs, Messages, Classes,
 Forms, Cgi, DB, DBTables;

type
 TForm1 = class(TForm)
  DataSource1: TDataSource;
  Table1: TTable;
  CGIEnvData1: TCGIEnvData;
  DataSource2: TDataSource;
  Table2: TTable;
  DataSource3: TDataSource;
  Table3: TTable;
  DataSource4: TDataSource;
  Query1: TQuery;
  procedure FormCreate(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);

var
 ProviderID : string;
 TechID : string;
 TechName : string;
 tObjectType : string;
 tProblemArea : string;
 tFunctionalArea : string;
```

```
  tSolutionMethod : string;
  tIndType : string;
  tOrgType : string;
  tURL : string;
  ExcInd : string;
  PMessage : TStringList;
  CMessage : TStringList;

begin

  with CGIEnvData1 do
  begin
    { required when this program runs under WebSite}
    webSiteINIFilename := paramstr(1);
    application.onException := cgiErrorHandler;
    application.processMessages;

    {receive input fields from HTML form}
    ProviderID := getSmallField( 'ProviderID' );
    TechID := getSmallField( 'TechID' );
    TechName := getSmallField( 'TechName' );
    tObjectType := getSmallField( 'tObjectType' );
    tProblemArea := getSmallField( 'tProblemArea' );
    tFunctionalArea := getSmallField( 'tFunctionalArea' );
    tSolutionMethod := getSmallField( 'tSolutionMethod' );
    tIndType := getSmallField( 'tIndType' );
    tOrgType := getSmallField( 'tOrgType' );
    tURL := getSmallField( 'tURL' );
    ExcInd := getSmallField( 'ExcInd' );

    PMessage := TStringList.create;
    PMessage.clear;
    getTextArea( 'Purpose', PMessage );

    CMessage := TStringList.create;
    CMessage.clear;
    getTextArea( 'Comments', CMessage );

    {update Active Provider table}
    with Query1 do
     begin
       close;
       SQL.clear;
```

```
   sql.add('UPDATE ACT_PROV ');
   sql.add('SET LastActionTime = ''' +DateTimeToStr(Now)+ '''');
   sql.add('WHERE ActProvID = ''' + ProviderID + '''');
   ExecSQL;
  end;

{add record to Technology table}
Table1.open;
Table1.AppendRecord([ProviderID,TechID,TechName,tObjectType,tProblemArea,
            tFunctionalArea,tSolutionMethod,tIndType,tOrgType,
            tURL,ExcInd]);
Table1.close;

{add record to Technology Mirror table}
Table2.open;
Table2.AppendRecord([ProviderID,TechID,TechName,tObjectType,tProblemArea,
            tFunctionalArea,tSolutionMethod,tIndType,tOrgType,
            tURL,ExcInd]);
Table2.close;

{add record to Technology Info table}
Table3.open;
Table3.AppendRecord([ProviderID,TechID,PMessage,CMessage,
            DateTimeToStr(Now),DateTimeToStr(Now)]);
Table3.close;

{Standard HTML header information}
createStdout;
sendPrologue;

send( '<HTML><HEAD>' );
sendTitle( 'DecisionNet Technology Registration' );
send( '</HEAD><BODY BGCOLOR="80B7B0">' );
send('<center><h1>DecisionNet Technology Registration</h1>');

{Check if technology is Exclusive or Independent.  If exclusive, send
user to regexcl.exe; if independent, registration is complete}
if ExcInd = 'Exclusive' then
 begin
   send('<h2>Your Information has been accepted.  Please follow the ');
   send('link below to continue registration of your Exclusive ');
   send('Technology.</h2></center>');
```

```
      {capture ProviderID and TechID, send user to Exclusive Tech Reg Form}
      send('<CENTER>');
      send('<FORM method=post action="'+
       'http://131.120.39.63/cgi-win/dnet/provider/null.exe">');
      send('<input type="hidden" name="ProviderID" '+
       'value="'+ProviderID+'"></td>');
      send('<input type="hidden" name="TechID" '+
       'value="'+TechID+'"></td>');
      send('<input type="submit" value="Exclusive Tech Registration">');
      send('</form>');
      send('</CENTER>');
     end

   else
    begin
     send('<h2>Your technology is now registered. Thank you for your input.');
     send('</h2></center>');
     send('<p>');

     {capture ProviderID, send user to Provider Menu}
     send('<CENTER>');
     send('<FORM method=post action="'+
      'http://131.120.39.63/cgi-win/dnet/provider/menu.exe">');
     send('<input type="hidden" name="ProviderID" '+
      'value="'+ProviderID+'"></td>');
     send('<input type="submit" value="Registered Provider Menu">');
     send('</form>');
     send('</CENTER>');
    end;

   {Standard HTML footer information}
   send('<p>');
   sendHR;
   send('<p><font size=-1><i>');
   send( 'This application was created by Steve Earley for Professor ');
   send( 'Hemant Bhargava.<br>' );
   send( 'Generated on ' + webdate(now) );
   send( '</i></font></BODY></HTML>' );
   closeStdout;
  end;
end;
end.
```

5. Update Technology Scripts

```
unit Updttec1; {Technology Update; created by Steve Earley;
        updated: 4 Jun 96.}

interface

uses
 SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
 Forms, Dialogs, Cgi, DBTables, DB;

type
 TForm1 = class(TForm)
  DataSource1: TDataSource;
  CGIEnvData1: TCGIEnvData;
  DataSource2: TDataSource;
  Table1: TTable;
  Query1: TQuery;
  procedure FormCreate(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);

var
 ProviderID : string;
 TechEntry : boolean;

begin
 with CGIEnvData1 do
 begin
  { required when this program runs under WebSite}
  webSiteINIFilename := paramstr(1);
```

```
application.onException := cgiErrorHandler;
application.processMessages;

{Get ProviderID field from menu.exe}
ProviderID := getSmallField( 'ProviderID' );

{update Active Provider table}
with Query1 do
 begin
  close;
  SQL.clear;
  sql.add('UPDATE ACT_PROV ');
  sql.add('SET LastActionTime = ''' +DateTimeToStr(Now)+ ''');
  sql.add('WHERE ActProvID = ''' + ProviderID + ''');
  ExecSQL;
 end;

{standard dynamic HTML header information}
createStdout;
sendPrologue;
send( '<HTML><HEAD>' );
sendTitle( 'Update DecisionNet Technology' );
send( '</HEAD><BODY BGCOLOR="80B7B0">' );
send('<center><h1>Update DecisionNet Technology</h1></center>');

with Table1 do    {Check Technolo.db to ensure technologies exist}
 begin
  open;
  TechEntry := FindKey([ProviderID]);
  close;
 end;

if TechEntry = False then   {No technologies exist for provider}
 begin
  send('<center><h2>Sorry, but you do not have any registered ');
  send('technologies for <I>' + ProviderID + '</I>.  You must register ');
  send('a technology before you can modify it.</h2>');
  send('<P><HR><P>');

  {Capture ProviderID, send user to Provider Menu}
  send('<FORM method=post action="'+
   'http://131.120.39.63/cgi-win/dnet/provider/menu.exe">');
  send('<input type="hidden" name="ProviderID" '+
```

```
    'value="'+ProviderID+'"></td>');
   send('<input type="submit" value="Return to Provider Menu">');
   send('</form></center>');
  end

else        {technologies exist for provider}
  begin
   send('Please enter your password in the field provided, choose ');
   send('the technology you would like to update, then press the  ');
   send('"Update Technology" button.  You will then be sent to the ');
   send('appropriate form for modifying your technology. ');
   send(' Thank you for using DecisionNet.');
   send('<P><HR><P>');

   send('<center>');
   send('<FORM method=post action=');
   send('"http://131.120.39.63/cgi-win/dnet/provider/updtteca.exe">');
   send('<input type="hidden" name="ProviderID" ');
   send('value="' +ProviderID+ '">');

   send('<TABLE BORDER=6 CELLPADDING=6>');

   send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
   send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
   send('<B>Password:</B>');
   send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
   send('<INPUT type="password" name="pPassword" size=10>');

   send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
   send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
   send('<B>Technology Name:</B>');
   send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
   send('<SELECT NAME="TechID" TYPE="text" SIZE=1>');

   with Table1 do    {Check Technolo.db for Option Entries}
    begin
     open;
     first;  {puts cursor on first record in table}

     while not EOF do
      begin
       if FieldByName('ProviderID').AsString = ProviderID then
        begin
```

```
        send('<OPTION VALUE= "'+FieldByName('TechID').AsString+'">');
        send(FieldByName('TechName').AsString);
       end;
       next; {puts cursor on next record}
     end;
    close;
   end;

  send('</SELECT></TD>');
  send('</TABLE>');

  send('<P><P>');
  send('<input type="submit" value= "Update Technology">');
  send('<input type="reset" value="Clear Form">');
  send('</form>');

  {If user cancels, capture ProviderID, send user to Provider Menu}
  send('<FORM method=post action="'+
   'http://131.120.39.63/cgi-win/dnet/provider/menu.exe">');
  send('<input type="hidden" name="ProviderID" '+
   'value="'+ProviderID+'"></td>');
  send('<input type="submit" value="Cancel, Return to Provider Menu">');
  send('</form>');
  send('</CENTER>');
 end;

{standard dynamic HTML footer information}
send('<p>');
sendHR;
send('<p><i><font size=-1>');
send( 'This application was created by Steve Earley ');
send( 'for Professor Hemant Bhargava.<br>' );
send( 'Generated on ' + webdate(now) );
send( '</i></font></BODY></HTML>' );
closeStdout;

 end;
end;
end.
```

```
unit Updteca1; {Technology Update Form; created by Steve Earley;
        updated 20 Jun 96.}

interface

uses
 SysUtils, WinTypes, WinProcs, Messages, Classes,
 Forms, Cgi, DB, DBTables, Cgidb;

type
 TForm1 = class(TForm)
  DataSource1: TDataSource;
  CGIEnvData1: TCGIEnvData;
  DataSource2: TDataSource;
  Table1: TTable;
  DataSource3: TDataSource;
  Table2: TTable;
  Query1: TQuery;
  DataSource4: TDataSource;
  Table3: TTable;
  CGIDB1: TCGIDB;
  DataSource5: TDataSource;
  Table4: TTable;
  procedure FormCreate(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);

var
 ProviderID : string;
 pPassword : string;
 TechID : string;
```

```
begin
 with CGIEnvData1 do
 begin
   { required when this program runs under WebSite}
   webSiteINIFilename := paramstr(1);
   application.onException := cgiErrorHandler;
   application.processMessages;

   {receive input fields from updttech.exe}
   ProviderID := getSmallField( 'ProviderID' );
   pPassword := getSmallField( 'pPassword' );
   TechID := getSmallField( 'TechID' );

   {Standard Header Information}
   createStdout;
   sendPrologue;

   send( '<HTML><HEAD>' );
   sendTitle( 'DecisionNet Technology Update Form' );
   send( '</HEAD><BODY BGCOLOR="80B7B0">' );

   with Table1 do   {search for user in Provider table;
                     GotoKey returns True if already logged in}
     begin
       open;
       SetKey;
       FieldByName('ProviderID').AsString := ProviderID;
       GoToKey;
     end;

   if (Table1.FieldByName('pPassword').AsString <> pPassword) or
    (pPassword = cginotfound) then   {Password does not match ProviderID, or
                                      Password not entered}

     begin
       send('<center><h1>Incorrect Attempt!</h1>');
       send('<h2>Please verify that your Password ' +
         'is correct, then try again.</h2></center>');
       send('<p>');
       sendHR;
       send('<p>');
```

```
    {capture ProviderID, send user back to Initial Update Form}
    send('<CENTER>');
    send('<FORM method=post action="'+
      'http://131.120.39.63/cgi-win/dnet/provider/updttech.exe">');
    send('<input type="hidden" name="ProviderID" '+
       'value="'+ProviderID+'"></td>');
    send('<input type="submit" value="Back to Tech Update Page">');
    send('</form>');
    send('</CENTER>');
  end

else  {ProviderID matches pPassword}
  begin
   {update Active Provider table}
   with Query1 do
    begin
     close;
     SQL.clear;
     sql.add('UPDATE ACT_PROV ');
     sql.add('SET LastActionTime = ''' +DateTimeToStr(Now)+ '''');
     sql.add('WHERE ActProvID = ''' + ProviderID + '''');
     ExecSQL;
    end;

   with Table3 do  {Search Technology Table for record to be modified}
    begin
     open;
     FindKey([ProviderID,TechID]);
    end;

   with Table4 do  {Search TechInfo Table for record to be modified}
    begin
     open;
     FindKey([ProviderID,TechID]);
    end;

   {Registration Form}
   send('<center><h1>Technology Update Form</h1></center>');
   send('Please fill in all of the following fields, then press the ');
   send('"Update" button. For each category, choose the term that ');
   send('best describes your technology.  These terms will later be used ');
   send('in an indexed search of technologies.  You cannot modify your ');
   send('Provider ID or your Technology ID number. ');
```

```
send('<center>');
send('<FORM method=post action=');
send('"http://131.120.39.63/cgi-win/dnet/provider/updttecb.exe">');
send('<input type="hidden" name="ProviderID" value="'+ProviderID+'">');
send('<input type="hidden" name="TechID" value="'+TechID+'">');

send('<TABLE BORDER=6 CELLPADDING=6>');

send('<TR><TD>Provider ID:');
send('</TD><td>' +ProviderID+ '</td>');

send('<TR><TD>Technology ID:');
send('</TD><td>' +TechID+ '</td>');

send('<TR><TD>Technology Name:');
send('</TD><td><input type="text" name="TechName" ');
send('size=20 maxlength=20 ');
send('value="'+Table3.FieldByName('TechName').AsString+'"></td>');

send('<TR><TD>Object Type:');
send('</td><td><SELECT NAME="tObjectType" TYPE="text" SIZE=1>');
send('<OPTION SELECTED ');
send('VALUE="'+Table3.FieldByName('tObjectType').AsString+'">');
send(Table3.FieldByName('tObjectType').AsString);

with Table2 do    {Check Taxonomy for Option Entries}
 begin
  open;
  first;  {puts cursor on first record in table}

  while not EOF do
   begin
    if FieldByName('Parent').AsString = 'Object Type' then
     begin
      send('<OPTION VALUE= "'+FieldByName('Child').AsString+'">');
      send(FieldByName('Child').AsString);
     end;
    next;  {puts cursor on next record}
   end;
  close;
 end;

send('</SELECT></TD>');
```

```
send('<TR><TD>Problem Area:');
send('</TD> <td> <SELECT NAME="tProblemArea" SIZE=1>');
send('<OPTION SELECTED ');
send('VALUE="'+Table3.FieldByName('tProblemArea').AsString+'">');
send(Table3.FieldByName('tProblemArea').AsString);

with Table2 do    {Check Taxonomy for Option Entries}
 begin
  open;
  first; {puts cursor on first record in table}

  while not EOF do
   begin
    if FieldByName('Parent').AsString = 'Problem Area' then
     begin
      send('<OPTION VALUE= "'+FieldByName('Child').AsString+'">');
      send(FieldByName('Child').AsString);
     end;
    next; {puts cursor on next record}
   end;
  close;
 end;

send('</SELECT></TD>');

send('<TR><TD>Functional Area:');
send('</td><td><SELECT NAME="tFunctionalArea" TYPE="text" SIZE=1>');
send('<OPTION SELECTED ');
send('VALUE="'+Table3.FieldByName('tFunctionalArea').AsString+'">');
send(Table3.FieldByName('tFunctionalArea').AsString);

with Table2 do    {Check Taxonomy for Option Entries}
 begin
  open;
  first; {puts cursor on first record in table}

  while not EOF do
   begin
    if FieldByName('Parent').AsString = 'Functional Area' then
     begin
      send('<OPTION VALUE= "'+FieldByName('Child').AsString+'">');
      send(FieldByName('Child').AsString);
     end;
```

```
  next; {puts cursor on next record}
 end;
close;
end;

send('</SELECT></TD>');

send('<TR><TD>Solution Method:');
send('</TD> <td><SELECT NAME="tSolutionMethod" SIZE=1>');
send('<OPTION SELECTED ');
send('VALUE="'+Table3.FieldByName('tSolutionMethod').AsString+'">');
send(Table3.FieldByName('tSolutionMethod').AsString);

with Table2 do    {Check Taxonomy for Option Entries}
 begin
 open;
 first; {puts cursor on first record in table}

 while not EOF do
  begin
   if FieldByName('Parent').AsString = 'Solution Method' then
    begin
     send('<OPTION VALUE= "'+FieldByName('Child').AsString+'">');
     send(FieldByName('Child').AsString);
    end;
   next; {puts cursor on next record}
  end;
 close;
end;

send('</SELECT></TD>');

send('<TR><TD>Industry Type:');
send('</TD> <td><SELECT NAME="tIndType" SIZE=1>');
send('<OPTION SELECTED ');
send('VALUE="'+Table3.FieldByName('tIndType').AsString+'">');
send(Table3.FieldByName('tIndType').AsString);

with Table2 do    {Check Taxonomy for Option Entries}
 begin
 open;
 first; {puts cursor on first record in table}
```

```
 while not EOF do
  begin
   if FieldByName('Parent').AsString = 'Industry Type' then
    begin
     send('<OPTION VALUE= "'+FieldByName('Child').AsString+'">');
     send(FieldByName('Child').AsString);
    end;
   next; {puts cursor on next record}
  end;
 close;
end;

send('</SELECT></TD>');

send('<tr><td>Organization Type:');
send('</td><td><SELECT NAME="tOrgType" SIZE=1>');
send('<OPTION SELECTED ');
send('VALUE="'+Table3.FieldByName('tOrgType').AsString+'">');
send(Table3.FieldByName('tOrgType').AsString);

with Table2 do    {Check Taxonomy for Option Entries}
 begin
  open;
  first; {puts cursor on first record in table}

  while not EOF do
   begin
    if FieldByName('Parent').AsString = 'Organization Type' then
     begin
      send('<OPTION VALUE= "'+FieldByName('Child').AsString+'">');
      send(FieldByName('Child').AsString);
     end;
    next; {puts cursor on next record}
   end;
  close;
 end;

send('</SELECT></TD>');

send('<TR><TD>URL:');
send('</TD><td><input type="text" name="tURL" ');
send('size=50 maxlength=255 ');
send('value="'+Table3.FieldByName('tURL').AsString+'"></td>');
```

```
send('<TR><TD>Technology Type:');
if Table3.FieldByName('ExcInd').AsString = 'Independent' then
 begin
  send('</TD><td><input type="radio" checked name="ExcInd" ');
  send('value="Independent"> Independent');
  send('<input type="radio" name="ExcInd" value="Exclusive"> ');
  send('Exclusive</td>');
 end
else
 begin
  send('</TD><td><input type="radio" name="ExcInd" ');
  send('value="Independent"> Independent');
  send('<input type="radio" checked name="ExcInd" value="Exclusive"> ');
  send('Exclusive</td>');
 end;

send('<TR><TD>Purpose:</TD>');
send('<TD><textarea rows=5 cols=50 name="Purpose">');
CGIDB1.SendMemo(Table4.FieldByName('Purpose'));
send('</textarea>');

send('<TR><TD>Comments:</TD>');
send('<TD><textarea rows=8 cols=50 name="Comments">');
CGIDB1.SendMemo(Table4.FieldByName('Comments'));
send('</textarea>');

send('</TABLE>');
send('<P>');
send('<input type="submit" value= "Update">');
send('<input type="reset" value="Clear Form"></center>');
send('</form>');
send('<P>');

Table1.close;
Table3.close;
```

{If user cancels, capture ProviderID, send user to Provider Menu}

```
send('<CENTER>');
send('<FORM method=post action="'+
 'http://131.120.39.63/cgi-win/dnet/provider/menu.exe">');
send('<input type="hidden" name="ProviderID" '+
 'value="'+ProviderID+'">');
send('<input type="submit" value="Cancel, Return to Provider Menu">');
```

```
      send('</form>');
      send('</CENTER>');
    end;

  {Standard footer information}
  sendHR;
  send('<p><I><font size=-1>');
  send( 'This application was created by Steve Earley for Professor ');
  send( 'Hemant Bhargava.<br>' );
  send( 'Generated on ' + webdate(now) );
  send( '</i></font></BODY></HTML>' );
  closeStdout;

 end;
end;
end.
```

```
unit Updtecb1; {Technology Update Script; created by Steve Earley;
        updated 7 Jun 96.}

interface

uses
 SysUtils, WinTypes, WinProcs, Messages, Classes,
 Forms, Cgi, DB, DBTables, Cgidb;

type
 TForm1 = class(TForm)
  DataSource1: TDataSource;
  CGIEnvData1: TCGIEnvData;
  DataSource2: TDataSource;
  DataSource3: TDataSource;
  DataSource4: TDataSource;
  Query1: TQuery;
  Query2: TQuery;
  Query3: TQuery;
  Table1: TTable;
  procedure FormCreate(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);

var
 ProviderID : string;
 TechID : string;
 TechName : string;
 tObjectType : string;
 tProblemArea : string;
 tFunctionalArea : string;
```

```
  tSolutionMethod : string;
  tIndType : string;
  tOrgType : string;
  tURL : string;
  ExcInd : string;
  PMessage : TStringList;
  CMessage : TStringList;
  TempDate : string;

begin
  with CGIEnvData1 do
  begin
    { required when this program runs under WebSite}
    webSiteINIFilename := paramstr(1);
    application.onException := cgiErrorHandler;
    application.processMessages;

    {receive input fields from HTML form}
    ProviderID := getSmallField( 'ProviderID' );
    TechID := getSmallField( 'TechID' );
    TechName := getSmallField( 'TechName' );
    tObjectType := getSmallField( 'tObjectType' );
    tProblemArea := getSmallField( 'tProblemArea' );
    tFunctionalArea := getSmallField( 'tFunctionalArea' );
    tSolutionMethod := getSmallField( 'tSolutionMethod' );
    tIndType := getSmallField( 'tIndType' );
    tOrgType := getSmallField( 'tOrgType' );
    tURL := getSmallField( 'tURL' );
    ExcInd := getSmallField( 'ExcInd' );

    PMessage := TStringList.create;
    PMessage.clear;
    getTextArea( 'Purpose', PMessage );

    CMessage := TStringList.create;
    CMessage.clear;
    getTextArea( 'Comments', CMessage );

    {update Active Provider table}
    with Query1 do
     begin
       close;
       SQL.clear;
```

```
  sql.add('UPDATE ACT_PROV ');
  sql.add('SET LastActionTime = ''' +DateTimeToStr(Now)+ '''');
  sql.add('WHERE ActProvID = ''' + ProviderID + '''');
  ExecSQL;
 end;

{Update Technology table}
with Query2 do
 begin
  close;
  SQL.clear;
  sql.add('UPDATE TECHNOLO ');
  sql.add('SET TechName = ''' +TechName+ '''');
  sql.add(', tObjectType = ''' +tObjectType+ '''');
  sql.add(', tProblemArea = ''' +tProblemArea+ '''');
  sql.add(', tFunctionalArea = ''' +tFunctionalArea+ '''');
  sql.add(', tSolutionMethod = ''' +tSolutionMethod+ '''');
  sql.add(', tIndType = ''' +tIndType+ '''');
  sql.add(', tOrgType = ''' +tOrgType+ '''');
  sql.add(', tURL = ''' +tURL+ '''');
  sql.add(', ExcInd = ''' +ExcInd+ '''');
  sql.add('WHERE (ProviderID = ''' + ProviderID + ''') ');
  sql.add('AND (TechID = ''' + TechID + ''')');
  ExecSQL;
 end;

{Update Technology Mirror table}
with Query3 do
 begin
  close;
  SQL.clear;
  sql.add('UPDATE TECHMIRR ');
  sql.add('SET TechName = ''' +TechName+ '''');
  sql.add(', tObjectType = ''' +tObjectType+ '''');
  sql.add(', tProblemArea = ''' +tProblemArea+ '''');
  sql.add(', tFunctionalArea = ''' +tFunctionalArea+ '''');
  sql.add(', tSolutionMethod = ''' +tSolutionMethod+ '''');
  sql.add(', tIndType = ''' +tIndType+ '''');
  sql.add(', tOrgType = ''' +tOrgType+ '''');
  sql.add(', tURL = ''' +tURL+ '''');
  sql.add(', ExcInd = ''' +ExcInd+ '''');
  sql.add('WHERE (ProviderID = ''' + ProviderID + ''') ');
  sql.add('AND (TechID = ''' + TechID + ''')');
```

```
  ExecSQL;
 end;

{Update Technology Info table: requires a delete and an append to work
with the memo fields}
with Table1 do
 begin
  open;
  FindKey([ProviderID,TechID]);
  TempDate := FieldByName('DateRegistered').AsString;
  delete;
  AppendRecord([ProviderID,TechID,PMessage,CMessage,
          TempDate,DateTimeToStr(Now)]);
  close;
 end;

{Standard HTML header information}
createStdout;
sendPrologue;

send( '<HTML><HEAD>' );
sendTitle( 'DecisionNet Technology Update' );
send( '</HEAD><BODY BGCOLOR="80B7B0">' );
send('<center><h1>DecisionNet Technology Update</h1>');

{Check if technology is Exclusive or Independent.  If exclusive, send
user to updtexcl.exe; if independent, update is complete}
if ExcInd = 'Exclusive' then
 begin
  send('<h2>Your Information has been accepted.  Please follow the ');
  send('link below to continue updating your Exclusive ');
  send('Technology.</h2></center>');

  {capture ProviderID and TechID, send user to Exclusive Tech Update Form}
  send('<CENTER>');
  send('<FORM method=post action="'+
   'http://131.120.39.63/cgi-win/dnet/provider/null.exe">');
  send('<input type="hidden" name="ProviderID" '+
   'value="'+ProviderID+'"></td>');
  send('<input type="hidden" name="TechID" '+
   'value="'+TechID+'"></td>');
  send('<input type="submit" value="Exclusive Tech Modification">');
  send('</form>');
```

```
      send('</CENTER>');
    end

  else
    begin
      send('<h2>Your technology"s information has been modified. ');
      send(' Thank you for your input.');
      send('</h2></center>');
      send('<p>');

      {capture ProviderID, send user to Provider Menu}
      send('<CENTER>');
      send('<FORM method=post action="'+
        'http://131.120.39.63/cgi-win/dnet/provider/menu.exe">');
      send('<input type="hidden" name="ProviderID" '+
        'value="'+ProviderID+'"></td>');
      send('<input type="submit" value="Registered Provider Menu">');
      send('</form>');
      send('</CENTER>');
    end;

  {Standard HTML footer information}
  send('<p>');
  sendHR;
  send('<p><font size=-1><i>');
  send( 'This application was created by Steve Earley for Professor ');
  send( 'Hemant Bhargava.<br>' );

  send( 'Generated on ' + webdate(now) );

  send( '</i></font></BODY></HTML>' );
  closeStdout;

 end;
end;
end.
```

6. Technology Information

```
unit Techinf1;  {Provider Technology Usage script; created by Steve Earley,
                 last update 11 Jun 96.}

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, DB, Cgidb, Cgi, DBTables;

type
  TForm1 = class(TForm)
    CGIEnvData1: TCGIEnvData;
    CGIDB1: TCGIDB;
    DataSource1: TDataSource;
    Query1: TQuery;
    DataSource2: TDataSource;
    Query2: TQuery;
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}


procedure TForm1.FormCreate(Sender: TObject);

var
  ProviderID : string;

begin
     with CGIEnvData1 do
     begin
```

```
{required when this program runs under WebSite}
    websiteINIFilename := paramstr(1);
    application.onException := cgiErrorHandler;
    application.processMessages;
    createStdout;
    sendPrologue;

{HTML page header info}
    send( '<HTML><HEAD>' );
    sendTitle( 'Technology Usage' );
    send( '</HEAD><BODY BGCOLOR="80B7B0">' );

{receive input field from Provider Menu; if user tries to go directly to
program, or tries to view records as a guest, raise an error}
ProviderID := getSmallField( 'ProviderID' );

 if (ProviderID = CGINotFound) or (ProviderID = 'guest') then
  begin
   send('<center><H1>Unable to View Records</h1>');
   send('<h2>You cannot view your information unless you have logged ');
   send('in as a valid Provider other than <I>'+ProviderID+'</I>. ');
   send('Please <a href="http://131.120.39.66/start.htm">login</a> ');
   send('to continue.</h2></center>');
  end

 else
  begin

    send('<center><H1>DecisionNet Technology Usage</h1></center>');
    send('This is a complete listing of all registered users who ');
    send('have accessed your technologies. ');
    send('<p>');

    {update Active Provider table}
    with Query1 do
       begin
          close;
          SQL.clear;
          sql.add('UPDATE ACT_PROV ');
          sql.add('SET LastActionTime = ''' +DateTimeToStr(Now)+ '''');
          sql.add('WHERE ActProvID = ''' + ProviderID + '''');
          ExecSQL;
       end;
```

```
    {Create output table using query on Used Technology table}
    with query2 do
       begin
          close;
          SQL.clear;
          SQL.add('SELECT ProviderID,TechID,UserID,LastActionTime ');
          SQL.add('FROM USEDTECH ');
          SQL.add('WHERE ProviderID = ''' + ProviderID + '''');
          open;
       end;

    send('<center>');
    CGIDB1.drawTable;
    query2.close;
    send('</center>');
    send('<p>');

  {Capture ProviderID, send user to Provider Menu}
  send('<CENTER>');
  send('<FORM method=post action="'+
    'http://131.120.39.63/cgi-win/dnet/provider/menu.exe">');
  send('<input type="hidden" name="ProviderID" '+
    'value="'+ProviderID+'"></td>');
  send('<input type="submit" value="Return to Provider Menu">');
  send('</form>');
  send('</CENTER>');
   end;

  {HTML page footer info}
  sendHR;
  send('<p><I><FONT SIZE=-1>');
  send( 'This application was created by Steve Earley ');
  send( 'for Professor Hemant Bhargava.<br>' );
  send( 'Generated on ' + webdate(now) );
  send( '</FONT></I></BODY></HTML>' );

  closeStdout;
  closeApp( application );

   end;
end;
end.
```

7. Withdraw Technology Scripts

```
unit Wdtech1; {Technology Withdraw; created by Steve Earley; updated: 4 Jun 96.}

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, Cgi, DBTables, DB;

type
  TForm1 = class(TForm)
    DataSource1: TDataSource;
    CGIEnvData1: TCGIEnvData;
    DataSource2: TDataSource;
    Table1: TTable;
    Query1: TQuery;
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);

var
  ProviderID : string;
  TechEntry : boolean;

begin
  with CGIEnvData1 do
  begin
    { required when this program runs under WebSite}
    webSiteINIFilename := paramstr(1);
    application.onException := cgiErrorHandler;
```

```
application.processMessages;

{Get ProviderID field from menu.exe}
ProviderID := getSmallField( 'ProviderID' );

{update Active Provider table}
with Query1 do
 begin
  close;
  SQL.clear;
  sql.add('UPDATE ACT_PROV ');
  sql.add('SET LastActionTime = ''' +DateTimeToStr(Now)+ '''');
  sql.add('WHERE ActProvID = ''' + ProviderID + '''');
  ExecSQL;
 end;

{standard dynamic HTML header information}
createStdout;
sendPrologue;
send( '<HTML><HEAD>' );
sendTitle( 'Withdraw DecisionNet Technology' );
send( '</HEAD><BODY BGCOLOR="80B7B0">' );
send('<center><h1>Withdraw DecisionNet Technology</h1></center>');

with Table1 do    {Check Technolo.db to ensure technologies exist}
 begin
  open;
  TechEntry := FindKey([ProviderID]);
  close;
 end;

if TechEntry = False then   {No technologies exist for provider}
 begin
  send('<center><h2>Sorry, but you do not have any registered ');
  send('technologies for <I>' + ProviderID + '</I>.  You must register ');
  send('a technology before you can withdraw it.</h2>');
  send('<P><HR><P>');

  {Capture ProviderID, send user to Provider Menu}
  send('<FORM method=post action="'+
   'http://131.120.39.63/cgi-win/dnet/provider/menu.exe">');
  send('<input type="hidden" name="ProviderID" '+
   'value="'+ProviderID+'"></td>');
```

```
  send('<input type="submit" value="Return to Provider Menu">');
  send('</form></center>');
 end

else        {technologies exist for provider}
 begin
 send('Please enter your password in the field provided, choose ');
 send('the technology you would like to withdraw, then press the  ');
 send('"Withdraw Technology" button.  Should you choose to re-register ');
 send('this technology at a later date, it must be registered under a ');
 send('different Technology ID number. ');
 send(' Thank you for using DecisionNet.');
 send('<P><HR><P>');

 send('<center>');
 send('<FORM method=post action=');
 send('"http://131.120.39.63/cgi-win/dnet/provider/wdtecha.exe">');
 send('<input type="hidden" name="ProviderID" ');
 send('value="' +ProviderID+ '">');

 send('<TABLE BORDER=6 CELLPADDING=6>');

 send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
 send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
 send('<B>Password:</B>');
 send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
 send('<INPUT type="password" name="pPassword" size=10>');

 send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
 send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
 send('<B>Technology Name:</B>');
 send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
 send('<SELECT NAME="TechID" TYPE="text" SIZE=1>');

 with Table1 do    {Check Technolo.db for Option Entries}
  begin
  open;
  first; {puts cursor on first record in table}

  while not EOF do
   begin
    if FieldByName('ProviderID').AsString = ProviderID then
     begin
```

```
        send('<OPTION VALUE= "'+FieldByName('TechID').AsString+'">');
        send(FieldByName('TechName').AsString);
      end;
       next; {puts cursor on next record}
    end;
   close;
  end;

  send('</SELECT></TD>');
  send('</TABLE>');

  send('<P><P>');
  send('<input type="submit" value= "Withdraw Technology">');
  send('<input type="reset" value="Clear Form">');
  send('</form>');

  {If user cancels, capture ProviderID, send user to Provider Menu}
  send('<FORM method=post action="'+
   'http://131.120.39.63/cgi-win/dnet/provider/menu.exe">');
  send('<input type="hidden" name="ProviderID" '+
   'value="'+ProviderID+'"></td>');
  send('<input type="submit" value="Cancel, Return to Provider Menu">');
  send('</form>');
  send('</CENTER>');
  end;

 {standard dynamic HTML footer information}
 send('<p>');
 sendHR;
 send('<p><i><font size=-1>');
 send( 'This application was created by Steve Earley ');
 send( 'for Professor Hemant Bhargava.<br>' );
 send( 'Generated on ' + webdate(now) );
 send( '</i></font></BODY></HTML>' );
 closeStdout;

 end;
end;
end.
```

```
unit Wdtecha1; {Technology Withdraw; created by Steve Earley;
        updated: 29 May 96.}

interface

uses
 SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
 Forms, Dialogs, Cgi, DBTables, DB;

type
 TForm1 = class(TForm)
  DataSource1: TDataSource;
  Table1: TTable;
  CGIEnvData1: TCGIEnvData;
  DataSource2: TDataSource;
  Query1: TQuery;
  DataSource3: TDataSource;
  Query2: TQuery;
  DataSource4: TDataSource;
  Query3: TQuery;
  procedure FormCreate(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);

var
 ProviderID : string;
 pPassword : string;
 TechID : string;

begin
```

```
with CGIEnvData1 do
begin
  { required when this program runs under WebSite}
  webSiteINIFilename := paramstr(1);
  application.onException := cgiErrorHandler;
  application.processMessages;

  {Get ProviderID field from menu.exe}
  ProviderID := getSmallField( 'ProviderID' );
  pPassword := getSmallField( 'pPassword' );
  TechID := getSmallField( 'TechID' );

  {standard dynamic HTML header information}
  createStdout;
  sendPrologue;
  send( '<HTML><HEAD>' );
  sendTitle( 'Withdraw DecisionNet Technology' );
  send( '</HEAD><BODY BGCOLOR="80B7B0">' );

  with Table1 do   {search for user in Provider table;
                    GotoKey returns True if already logged in}
    begin
      open;
      SetKey;
      FieldByName('ProviderID').AsString := ProviderID;
      GoToKey;
    end;

  if (Table1.FieldByName('pPassword').AsString <> pPassword) or
   (pPassword = cginotfound) then   {Password does not match ProviderID, or
                                     Password not entered}

    begin
      send('<center><h1>Incorrect Attempt!</h1>');
      send('<h2>Please verify that your Password ' +
        'is correct, then try again.</h2></center>');
      send('<p>');
      sendHR;
      send('<p>');

      {capture ProviderID, send user back to Withdrawl Form}
      send('<CENTER>');
      send('<FORM method=post action="'+
```

```
      'http://131.120.39.63/cgi-win/dnet/provider/wdtech.exe">');
    send('<input type="hidden" name="ProviderID" '+
      'value="'+ProviderID+'"></td>');
    send('<input type="submit" value="Back to Tech Withdrawl Page">');
    send('</form>');
    send('</CENTER>');
  end

else {ProviderID matches pPassword}
  begin
      {update Active Provider table}
      with Query1 do
       begin
        close;
        SQL.clear;
        sql.add('UPDATE ACT_PROV ');
        sql.add('SET LastActionTime = ''' +DateTimeToStr(Now)+ '''');
        sql.add('WHERE ActProvID = ''' + ProviderID + '''');
        ExecSQL;
       end;

      {Delete Technology from Technology table}
      with Query2 do
       begin
        close;
        SQL.clear;
        sql.add('DELETE FROM TECHNOLO ');
        sql.add('WHERE (TechID = ''' + TechID + ''') ');
        sql.add('AND (ProviderID = ''' + ProviderID + ''')');
        ExecSQL;
       end;

      {Delete Technology from TechInfo table}
      with Query3 do
       begin
        close;
        SQL.clear;
        sql.add('DELETE FROM TECHINFO ');
        sql.add('WHERE (TechID = ''' + TechID + ''') ');
        sql.add('AND (ProviderID = ''' + ProviderID + ''')');
        ExecSQL;
       end;
```

```
      send('<center><h1>Withdraw DecisionNet Technology</h1>');
      send('<HR><P><H2>Your technology is now removed from the system. ');
      send(' Thank you for using DecisionNet.</H2></center>');

      {capture ProviderID, send user to Provider Menu}
      send('<CENTER>');
      send('<FORM method=post action="'+
       'http://131.120.39.63/cgi-win/dnet/provider/menu.exe">');
      send('<input type="hidden" name="ProviderID" '+
       'value="'+ProviderID+'">');
      send('<input type="submit" value="Registered Provider Menu">');
      send('</form>');
      send('</CENTER>');
    end;

  Table1.close;

  {standard dynamic HTML footer information}
  send('<p>');
  sendHR;
  send('<p><i><font size=-1>');
  send( 'This application was created by Steve Earley ');
  send( 'for Professor Hemant Bhargava.<br>' );
  send( 'Generated on ' + webdate(now) );
  send( '</i></font></BODY></HTML>' );
  closeStdout;

 end;
end;
end.
```

8. Modify Provider Information

```
unit Modify1;  {Modify Provider Information; created by Steve Earley.
          Last Updated on 10 Jun 96.}

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, Cgi, DB, DBTables;

type
  TForm1 = class(TForm)
    CGIEnvData1: TCGIEnvData;
    DataSource1: TDataSource;
    Query1: TQuery;
    DataSource2: TDataSource;
    Table1: TTable;
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);

var
  ProviderID : string;

begin
  with CGIEnvData1 do
  begin
    {required when this program runs under WebSite}
    webSiteINIFilename := paramstr(1);
    application.onException := cgiErrorHandler;
```

```
application.processMessages;

{HTML page header info}
createStdout;
sendPrologue;
send( '<HTML><HEAD>' );
sendTitle( 'Modify Provider Information' );
send( '</HEAD><BODY BGCOLOR="80B7B0">' );

{receive input field from Provider Menu; if user tries to go directly to
program, or tries to register technology as a guest, raise an error}
ProviderID := getSmallField( 'ProviderID' );

 if (ProviderID = CGINotFound) or (ProviderID = 'guest') then
  begin
   send('<center><H1>Unable to Modify Record</h1>');
   send('<h2>You cannot modify your information unless you have logged ');
   send('in as a valid Provider other than <I>'+ProviderID+'</I>. ');
   send('Please <a href="http://131.120.39.66/start.htm">login</a> ');
   send('to continue.</h2></center>');
  end

 else
  begin

    {update Active Provider table}
    with Query1 do
       begin
          close;
          SQL.clear;
          sql.add('UPDATE ACT_PROV ');
          sql.add('SET LastActionTime = ''' +DateTimeToStr(Now)+ '''');
          sql.add('WHERE ActProvID = ''' + ProviderID + '''');
          ExecSQL;
       end;

with Table1 do    {puts cursor on correct record in Provider table}
    begin
       open;
       SetKey;
       FieldByName('ProviderID').AsString := ProviderID;
       GotoKey;
    end;
```

{Modification form}

```
send('<CENTER><h1>Modify Provider Information</h1></CENTER> ');

send('<B>Directions:</B>');

send('<OL><LI>Please enter your current password in the "Old Password" ');
send('field (even if you are not changing your password).');

send('<LI>For the remaining fields, fill in any information that ');
send('you want to change, then press the "Modify Record" button. If you ');
send('do not want to change the information as shown, please do not ');
send('delete the entries in these fields.');

send('<LI>You cannot change your Provider ID using this page.  If you ');
send('need to change your Provider ID, please feel free to contact us.');

send('<LI>If you change your password, ');
send('you must type in all three password fields (old, new, and new ');
send('again) for the change to take effect. ');

send('<center> ');
send('<FORM method=post action= ');
  send('"http://131.120.39.63/cgi-win/dnet/provider/modifya.exe"> ');
send('<input type="hidden" name="ProviderID" value="' +ProviderID+ '">');
send('<TABLE BORDER=6 CELLPADDING=6> ');
send('<td>Old Password:</td><td><input type="password" ');
  send('name="pPassword" size=10></td> ');
send('<TR><TD>New Password:</TD><TD><input type="password" ');
  send('name="pPassword2"size=10> </td> ');
send('<TR><TD>Re-type New Password:</TD><TD><input type="password" ');
  send('name="pPassword3"  size=10> </td> ');
send('<TR><TD> Name:</TD> <td> <input type="text" ');
  send('name="pName" size=30 value=');
  send('"' +Table1.FieldByName('pName').AsString+ '"></td> ');
send('<TR><TD>Home Page URL:</TD> <td><input type="text" ');
  send('name="pURL" size=50 value=');
  send('"' +Table1.FieldByName('pURL').AsString+ '"></td> ');
send('<TR><TD>  Email Address: </TD> <td><input type="text" ');
  send('name="pEMailAddress" size=50 value=');
  send('"' +Table1.FieldByName('pEMailAddress').AsString+ '"></td> ');
send('</TABLE> ');

send('<P>');
```

```
    send('<input type="submit" value= "Modify Record"> ');
    send('<input type="reset" value="Clear Form">');
    send('</form> ');

    {If user cancels, capture ProviderID, send user to Provider Menu}
    send('<FORM method=post action="'+
      'http://131.120.39.63/cgi-win/dnet/provider/menu.exe">');
    send('<input type="hidden" name="ProviderID" '+
      'value="'+ProviderID+'"></td>');
    send('<input type="submit" value="Cancel, Return to Provider Menu">');
    send('</form>');
    send('</CENTER>');
     end;

    {HTML page footer info}
    send('<HR><I><FONT SIZE=-1>');
    send( 'This application was created by Steve Earley for Professor ');
    send( 'Hemant Bhargava.<br>' );
    send( 'Generated on ' + webdate(now) );
    send( '</I></FONT></BODY></HTML>' );
    closeStdout;

  end;
end;
end.
```

```
unit Modifya1; {Provider Info Modification; created by Steve Earley;
        last update 20 May 96.}

interface

uses
 SysUtils, WinTypes, WinProcs, Messages, Classes,
 Forms, Cgi, DB, DBTables;

type
 TForm1 = class(TForm)
  DataSource1: TDataSource;
  Table1: TTable;
  CGIEnvData1: TCGIEnvData;
  DataSource2: TDataSource;
  Query1: TQuery;
  DataSource3: TDataSource;
  Query2: TQuery;
  DataSource4: TDataSource;
  Query3: TQuery;
  procedure FormCreate(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);

var
 ProviderID : string;
 pPassword : string;
 pPassword2 : string;
 pPassword3 : string;
 pName : string;
 pURL : string;
```

```
  pEMailAddress : string;

begin
  with CGIEnvData1 do
  begin
    {required when this program runs under WebSite}
    webSiteINIFilename := paramstr(1);
    application.onException := cgiErrorHandler;
    application.processMessages;

    {receive input fields from HTML form generated by modify.exe}
    ProviderID := getSmallField( 'ProviderID' );
    pPassword := getSmallField( 'pPassword' );
    pPassword2 := getSmallField( 'pPassword2' );
    pPassword3 := getSmallField( 'pPassword3' );

    if (pPassword2 = '') and (pPassword3 = '') then
      begin
        pPassword2 := pPassword;
        pPassword3 := pPassword;
       end;

    pName := getSmallField( 'pName' );
    pURL := getSmallField( 'pURL' );
    pEMailAddress := getSmallField( 'pEMailAddress' );

    {HTML page header info}
    createStdout;
    sendPrologue;
    send( '<HTML><HEAD>' );
    sendTitle( 'Modify Provider Information' );
    send( '</HEAD><BODY BGCOLOR="80B7B0">' );

    {Update Active Provider table}
    with Query1 do
      begin
        close;
        SQL.clear;
        sql.add('UPDATE ACT_PROV ');
        sql.add('SET LastActionTime = ''' +DateTimeToStr(Now)+ '''');
        sql.add('WHERE ActProvID = ''' + ProviderID + '''');
        ExecSQL;
      end;
```

```
with Table1 do   {puts cursor on correct record in Provider table}
   begin
      open;
      SetKey;
      FieldByName('ProviderID').AsString := ProviderID;
      GotoKey;
   end;

if (Table1.FieldByName('pPassword').AsString <> pPassword) or
 (pPassword = cginotfound) then   {Password does not match ProviderID, or
                        Password not entered}

  begin
    send('<center><h1>Incorrect Attempt!</h1>');
    send('<h2>Please verify that your Old Password ' +
      'is correct, then try again.</h2></center>');
    send('<p>');
    sendHR;
    send('<p>');

    {capture ProviderID, send user back to Modification Form}
    send('<CENTER>');
    send('<FORM method=post action="'+
      'http://131.120.39.63/cgi-win/dnet/provider/modify.exe">');
    send('<input type="hidden" name="ProviderID" '+
       'value="'+ProviderID+'"></td>');
    send('<input type="submit" value="Back to Modification Page">');
    send('</form>');
    send('</CENTER>');
  end

else  {ProviderID matches pPassword; ok to change information}
  begin
   if pPassword2 <> pPassword3 then {bad registration}
    begin
      send('<center><h1>Password Mismatch</h1>');
      send('<h2>Please verify your password choice, ');
      send('and try again.</h2></center>');
      send('<p>');
      sendHR;
      send('<p>');
```

```
    {capture ProviderID, send user back to Modification Form}
    send('<CENTER>');
    send('<FORM method=post action="'+
        'http://131.120.39.63/cgi-win/dnet/provider/modify.exe">');
    send('<input type="hidden" name="ProviderID" '+
        'value="'+ProviderID+'"></td>');
    send('<input type="submit" value="Back to Modification Page">');
    send('</form>');
    send('</CENTER>');
   end

  else  {correct sequence entered for changing password}
   begin
    pPassword := pPassword2;

{Update Provider table}
with Query2 do
     begin
     close;
     SQL.clear;
     sql.add('UPDATE PROVIDER ');
     sql.add('SET pPassword = ''' +pPassword+ '''');
     sql.add(', pName = ''' +pName+ '''');
     sql.add(', pURL = ''' +pURL+ '''');
     sql.add(', pEMailAddress = ''' +pEMailAddress+ '''');
     sql.add('WHERE ProviderID = ''' + ProviderID + '''');
     ExecSQL;
     end;

{Update Provider Mirror table}
with Query3 do
     begin
     close;
     SQL.clear;
     sql.add('UPDATE PROVMIRR ');
     sql.add('SET pPassword = ''' +pPassword+ '''');
     sql.add(', pName = ''' +pName+ '''');
     sql.add(', pURL = ''' +pURL+ '''');
     sql.add(', pEMailAddress = ''' +pEMailAddress+ '''');
     sql.add('WHERE ProviderID = ''' + ProviderID + '''');
     ExecSQL;
     end;
```

```
    {HTML page body for successful registration change}
    send('<center><h1>Provider Registration Changes Accepted</h1>');
    send('<h2>Your information has been updated for ' +
      'User Name <I>' + ProviderID  + '</I>. ');
    send(' Thank you for using DecisionNet.</h2></center>');
    send('<p>');
    sendHR;
    send('<p>');

    {capture ProviderID, send user to Provider Menu}
    send('<CENTER>');
    send('<FORM method=post action="'+
      'http://131.120.39.63/cgi-win/dnet/provider/menu.exe">');
    send('<input type="hidden" name="ProviderID" '+
      'value="'+ProviderID+'"></td>');
    send('<input type="submit" value="Registered Provider Menu">');
    send('</form>');
    send('</CENTER>');
    send('<p>');
    end;
    end;

  {HTML page footer info}
  send('<HR><p><i><font size=-1>');
  send( 'This application was created by Steve Earley for Professor ');
  send( 'Hemant Bhargava.<br>' );
  send( 'Generated on ' + webdate(now) );
  send( '</i></font></BODY></HTML>' );
  closeStdout;

end;
end;
end.
```

9. Provider Withdraw

```
unit Withdra1; {Provider Withdraw; created by Steve Earley; updated: 5 Jun 96.}

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, Cgi, DBTables, DB;

type
  TForm1 = class(TForm)
    DataSource1: TDataSource;
    Table1: TTable;
    CGIEnvData1: TCGIEnvData;
    DataSource2: TDataSource;
    Query1: TQuery;
    DataSource3: TDataSource;
    Query2: TQuery;
    DataSource4: TDataSource;
    Table2: TTable;
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);

var
  ProviderID : string;
  TechEntry : boolean;

begin
  with CGIEnvData1 do
```

```
begin
  { required when this program runs under WebSite}
  webSiteINIFilename := paramstr(1);
  application.onException := cgiErrorHandler;
  application.processMessages;

  {Get ProviderID field from menu.exe}
  ProviderID := getSmallField( 'ProviderID' );

  {standard dynamic HTML header information}
  createStdout;
  sendPrologue;
  send( '<HTML><HEAD>' );
  sendTitle( 'Withdraw as DecisionNet Provider' );
  send( '</HEAD><BODY BGCOLOR="80B7B0">' );

  with Table2 do    {Check Technolo.db to ensure no technologies exist
                for Provider}
   begin
    open;
    TechEntry := FindKey([ProviderID]);
    close;
   end;

 if TechEntry = False then  {No technologies exist for provider, so it is
                      ok to withdraw provider}
 begin
   with Table1 do   {search for user in Active Provider table;
                   GotoKey returns True if already logged in}
    begin
      open;
      SetKey;
      FieldByName('ActProvID').AsString := ProviderID;
      GoToKey;
    end;

   if Table1.GoToKey = True then
    begin
      {Delete user from Active Provider table}
      with Query1 do
        begin
         close;
         SQL.clear;
```

```
        sql.add('DELETE FROM ACT_PROV ');
        sql.add('WHERE ActProvID = ''' + ProviderID + '''');
        ExecSQL;
       end;
   end;

     {Delete user from Provider table}
     with Query2 do
       begin
        close;
        SQL.clear;
        sql.add('DELETE FROM PROVIDER ');
        sql.add('WHERE ProviderID = ''' + ProviderID + '''');
        ExecSQL;
       end;

  Table1.close;

  send('<center><h1>Withdraw as DecisionNet Provider</h1></center>');
  send('<HR><P>You are now removed from the system. ');
  send(' We are sorry to see you go. Please feel free to register ');
  send('with us again in the future. ');
  send(' We hope to have a fully-functional system by the end of June, ');
  send('and we invite your comments for any improvements or additions ');
  send('you would like to see. ');
  send(' Thank you for using DecisionNet.');

  send('<H3>For DSS or overall system questions, please contact:</H3>');
  send('<P><CENTER><A HREF="mailto:bhargava@nps.navy.mil">');
  send('Prof. Hemant Bhargava: <I>bhargava@nps.navy.mil</I></A></CENTER>');

  send('<H3>For technical questions and support, please contact:</H3>');
  send('<P><CENTER><A HREF="mailto:shearley@nps.navy.mil">');
  send('Steve Earley: <I>shearley@nps.navy.mil</A></CENTER>');

 end

 else  {Provider still has technologies registered, therefore provider
       cannot withdraw yet}
 begin
    send('<center><h1>Unable to Withdraw</h1></center>');

    send('<center><h2>Sorry, but you cannot withdraw as a registered ');
```

```
      send('provider until you withdraw all of the registered ');
      send('technologies for <I>' + ProviderID + '</I>.  Once you remove ');
      send('these technologies, you will be allowed to withdraw.</h2>');
      send('<P><HR><P>');

      {Capture ProviderID, send user to Provider Menu}
      send('<FORM method=post action="'+
       'http://131.120.39.63/cgi-win/dnet/provider/menu.exe">');
      send('<input type="hidden" name="ProviderID" '+
       'value="'+ProviderID+'"></td>');
      send('<input type="submit" value="Return to Provider Menu">');
      send('</form></center>');
    end;

    {standard dynamic HTML footer information}
    send('<p>');
    sendHR;
    send('<p><i><font size=-1>');
    send( 'This application was created by Steve Earley ');
    send( 'for Professor Hemant Bhargava.<br>' );
    send( 'Generated on ' + webdate(now) );
    send( '</i></font></BODY></HTML>' );
    closeStdout;

  end;
end;
end.
```

10. Provider Logout

```
unit Logout1; {Provider Logout; created by Steve Earley; updated: 20 May 96.}

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, Cgi, DBTables, DB;

type
  TForm1 = class(TForm)
    DataSource1: TDataSource;
    Table1: TTable;
    CGIEnvData1: TCGIEnvData;
    DataSource2: TDataSource;
    Query1: TQuery;
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);

var
  ProviderID : string;

begin
  with CGIEnvData1 do
  begin
    { required when this program runs under WebSite}
    webSiteINIFilename := paramstr(1);
    application.onException := cgiErrorHandler;
    application.processMessages;
```

```
{Get ProviderID field from menu.exe}
ProviderID := getSmallField( 'ProviderID' );

{standard dynamic HTML header information}
createStdout;
sendPrologue;
send( '<HTML><HEAD>' );
sendTitle( 'DecisionNet Provider Logout' );
send( '</HEAD><BODY BGCOLOR="80B7B0">' );

with Table1 do   {search for user in Active Provider table;
                  GotoKey returns True if already logged in}
  begin
    open;
    SetKey;
    FieldByName('ActProvID').AsString := ProviderID;
    GoToKey;
  end;

if Table1.GoToKey = True then
  begin
    {Delete user from Active Provider table}
    with Query1 do
      begin
       close;
       SQL.clear;
       sql.add('DELETE FROM ACT_PROV ');
       sql.add('WHERE ActProvID = ''' + ProviderID + '''');
       ExecSQL;
      end;
  end;

Table1.close;

send('<center><h1>DecisionNet Logout</h1></center>');
send('<HR><P>You are now logged out of the system. ');
send(' We hope to have a fully-functional system by the end of June, ');
send('and we invite your comments for any improvements or additions ');
send('you would like to see. ');
send(' Thank you for using DecisionNet.');

send('<H3>For DSS or overall system questions, please contact:</H3>');
send('<P><CENTER><A HREF="mailto:bhargava@nps.navy.mil">');
```

```
    send('Prof. Hemant Bhargava: <I>bhargava@nps.navy.mil</I></A></CENTER>');

    send('<H3>For technical questions and support, please contact:</H3>');
    send('<P><CENTER><A HREF="mailto:shearley@nps.navy.mil">');
    send('Steve Earley: <I>shearley@nps.navy.mil</A></CENTER>');

    {standard dynamic HTML footer information}
    send('<p>');
    sendHR;
    send('<p><i><font size=-1>');
    send( 'This application was created by Steve Earley ');
    send( 'for Professor Hemant Bhargava.<br>' );
    send( 'Generated on ' + webdate(now) );
    send( '</i></font></BODY></HTML>' );
    closeStdout;

  end;
end;
end.
```

11. List Technologies

```
unit Listtec1; {Provider Technology Listing; created by Steve Earley.
        Last updated 28 May 96.}

interface

uses
 SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
 Forms, Dialogs, DB, Cgidb, Cgi, DBTables;

type
 TForm1 = class(TForm)
  CGIEnvData1: TCGIEnvData;
  DataSource1: TDataSource;
  Query1: TQuery;
  DataSource2: TDataSource;
  Query2: TQuery;
  procedure FormCreate(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form1: TForm1;

implementation

{$R *.DFM}


procedure TForm1.FormCreate(Sender: TObject);

var
  sortkey : string;
  UserID : string;

begin
   with CGIEnvData1 do
   begin
      websiteINIFilename := paramstr(1);
```

```
application.onException := cgiErrorHandler;
application.processMessages;

sortkey := GetSmallField('sortkey');
UserID := GetSmallField('ProviderID');

{update Active Provider table}
with Query2 do
   begin
      close;
      SQL.clear;
      sql.add('UPDATE ACT_PROV ');
      sql.add('SET LastActionTime = "' +DateTimeToStr(Now)+ '"');
      sql.add('WHERE ActProvID = "' + UserID + '"');
      ExecSQL;
   end;

createStdout;
sendPrologue;

{standard header information}
send( '<HTML><HEAD>' );
sendTitle( 'List of Technologies' );
send( '</HEAD><BODY BGCOLOR="80B7B0">' );
send('<center><H1>DecisionNet Technologies</h1></center>');
send('These technologies are owned and maintained by their ');
send('individual providers.  DecisionNet"s purpose is to ');
send('facilitate access to these technologies.');
send('<p>');
send('<center>');

{build table of technologies using join of Technology and Provider}
with query1 do
 begin
    close;
    SQL.clear;
    SQL.add('SELECT TECHNOLO."TechName", PROVIDER."pName", ' +
        'TECHNOLO."tObjectType", TECHNOLO."tProblemArea", ' +
        'TECHNOLO."tURL", TECHNOLO."TechID", PROVIDER."ProviderID" ' +
        'FROM TECHNOLO,PROVIDER ' +
        'WHERE (TECHNOLO.ProviderID = PROVIDER.ProviderID)');
    SQL.add('ORDER BY '+sortkey+");
```

```
        open;
        first; {puts cursor on first record in table}

        send('<TABLE BORDER=6 CELLPADDING=6>');
        send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
        send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
        send('<B>Technology Name</B>');
        send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
        send('<B>Provider Name</B>');
        send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
        send('<B>Object Type</B>');
        send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
        send('<B>Problem Area</B>');

        while not EOF do
         begin
           send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
           send('<FORM METHOD = POST  ACTION= ');
           send('"http://131.120.39.63/cgi-win/dnet/provider/about.exe">');
           send('<INPUT TYPE=HIDDEN NAME="UserID" VALUE="'+UserID+'">');
           send('<INPUT TYPE=HIDDEN NAME="ProviderID" ');
           send('VALUE="'+ FieldByName('ProviderID').AsString +'">');
           send('<INPUT TYPE=HIDDEN NAME="TechID" ');
           send('VALUE="'+ FieldByName('TechID').AsString +'">');
           send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
        send('<INPUT TYPE=SUBMIT VALUE="'+ FieldbyName('TechName').AsString
+'">');
           send('</TD></FORM>');
           send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
           send(FieldByName('pName').AsString);
           send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
           send(FieldByName('tObjectType').AsString);
           send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
           send(FieldByName('tProblemArea').AsString);
           next; {puts cursor on next record}
         end;

        send('</TABLE>');
        close;

    end;

   send('</center>');
```

```
    send('<p>');
    sendHR;
    send('<p>');

    {capture UserID, send user to Provider Menu}
    send('<CENTER>');
    send('<FORM method=post action="'+
     'http://131.120.39.63/cgi-win/dnet/provider/menu.exe">');
    send('<input type="hidden" name="ProviderID" '+
     'value="'+UserID+'"></td>');
    send('<input type="submit" value="Registered Provider Menu">');
    send('</form>');
    send('</CENTER>');
     send('<p>');

     sendHR;
     send('<p><FONT SIZE=-1><I>');
     send( 'This application was created by Steve Earley');
     send( 'for Professor Hemant Bhargava.<br>' );
     send( '<P>' );
     send( 'Generated on ' + webdate(now) );
     send( '</I></FONT></BODY></HTML>' );

     closeStdout;
     closeApp( application );

      end;
end;
end.
```

12. Provider "About Technology" Script

```
unit About1;  {Provider "About Technology" script; created by Steve Earley.
              Last updated 11 Jun 96.}

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, DB, Cgidb, Cgi, DBTables;

type
  TForm1 = class(TForm)
    CGIEnvData1: TCGIEnvData;
    DataSource1: TDataSource;
    Query1: TQuery;
    DataSource2: TDataSource;
    Query2: TQuery;
    CGIDB1: TCGIDB;
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}


procedure TForm1.FormCreate(Sender: TObject);

var
    UserID : string;       {Person using the program}
    ProviderID : string;   {Provider of a given technology}
    TechID : string;       {TechID of a given technology}

begin
  with CGIEnvData1 do
```

```
begin
 websiteINIFilename := paramstr(1);
 application.onException := cgiErrorHandler;
 application.processMessages;
 createStdout;
 sendPrologue;

 {Get hidden fields from previous page}
 UserID := GetSmallField('UserID');
 ProviderID := GetSmallField('ProviderID');
 TechID := GetSmallField('TechID');

 {update Active Provider table}
 with Query1 do
  begin
   close;
   SQL.clear;
   sql.add('UPDATE ACT_PROV ');
   sql.add('SET LastActionTime = ''' +DateTimeToStr(Now)+ '''');
   sql.add('WHERE ActProvID = ''' + UserID + '''');
   ExecSQL;
  end;

 {build table of technologies using join of necessary tables}
 with query2 do
  begin
   close;
   SQL.clear;
   SQL.add('SELECT DISTINCT * ' +
        'FROM TECHNOLO,PROVIDER,TECHINFO ' +
        'WHERE (TECHNOLO.ProviderID = PROVIDER.ProviderID)' +
        'AND (TECHNOLO.TechID = TECHINFO.TechID)' +
        'AND (TECHNOLO.ProviderID = TECHINFO.ProviderID)' +
        'AND (TECHNOLO.ProviderID = '''+ ProviderID + ''')' +
        'AND (TECHNOLO.TechID = '''+ TechID + ''')');
   open;

  {standard header information}
  send( '<HTML><HEAD>' );
  sendTitle( 'DecisionNet Technology Details' );
  send( '</HEAD><BODY BGCOLOR="80B7B0">' );
  send('<center><H1>DecisionNet Technology Details</h1></center>');
```

```
send('This technology is owned and maintained by its ');
send('individual provider. DecisionNet''s purpose is to ');
send('facilitate access to this technology.');

if FieldByName('ExcInd').AsString = 'Independent' then
begin
send(' When you execute this technology, it will open into a new ');
send('browser window. It is important to close that window and ');
send('return to this frame then ');
send('click on the button below to go back to the Provider Menu. ');
send('This will allow DecisionNet ');
send('to keep track of certain information about you. If you ');
send('do not do this, you may be forced to log back in to access ');
send('DecisionNet again. Thank you.');
end;

send('<p>');
send('<center>');

send('<TABLE BORDER=6 CELLPADDING=6>');

send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<B>Technology Name:</B>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send(FieldByName('TechName').AsString);

send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<B>Provider Name:</B>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send(FieldByName('pName').AsString);

send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<B>Provider ID:</B>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send(FieldByName('ProviderID').AsString);

send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<B>Technology ID:</B>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
```

```
send(FieldByName('TechID').AsString);

send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<B>Registered:</B>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send(FieldByName('DateRegistered').AsString);

send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<B>Updated:</B>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send(FieldByName('LastUpdate').AsString);

send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<B>Object Type</B>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send(FieldByName('tObjectType').AsString);

send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<B>Problem Area</B>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send(FieldByName('tProblemArea').AsString);

send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<B>Functional Area</B>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send(FieldByName('tFunctionalArea').AsString);

send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<B>Solution Method</B>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send(FieldByName('tSolutionMethod').AsString);

send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<B>Industry Type</B>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send(FieldByName('tIndType').AsString);
```

```
send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<B>Organization Type</B>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send(FieldByName('tOrgType').AsString);
send('</TABLE>');
send('</center>');

send('<P>');
send('<B>Purpose:</B> ');
CGIDB1.SendMemo(FieldByName('Purpose'));
send('<P>');

send('<P>');
send('<B>Comments:</B> ');
CGIDB1.SendMemo(FieldByName('Comments'));
send('<P>');
send('<CENTER>');

if FieldByName('ExcInd').AsString = 'Independent' then
 begin
  send('<FORM method=post action="'+
     'http://131.120.39.63/cgi-win/dnet/provider/execind.exe" ');
  send('TARGET="execind">');
  send('<input type="hidden" name="UserID" '+
     'value="'+UserID+'">');
  send('<input type="hidden" name="ProviderID" '+
     'value="'+ProviderID+'">');
  send('<input type="hidden" name="TechID" '+
     'value="'+TechID+'">');
 end
else
 begin
  send('<FORM method=post action="'+
     'http://131.120.39.63/cgi-win/dnet/provider/null.exe">');
  send('<input type="hidden" name="ProviderID" '+
     'value="'+UserID+'">');
 end;

send('<input type="submit" value="Execute Technology">');
send('</form>');
send('</CENTER>');
```

```
        Query2.close;
        end;

    send('<p>');
    sendHR;
    send('<p>');

    {capture UserID, send user to Provider Menu}
    send('<CENTER>');
    send('<FORM method=post action="'+
     'http://131.120.39.63/cgi-win/dnet/provider/menu.exe">');
    send('<input type="hidden" name="ProviderID" '+
     'value="'+UserID+'">');
    send('<input type="submit" value="Registered Provider Menu">');
    send('</form>');
    send('</CENTER>');
    send('<p>');
    sendHR;

    send('<p><FONT SIZE=-1><I>');
    send( 'This application was created by Steve Earley');
    send( 'for Professor Hemant Bhargava.<br>' );
    send( 'Generated on ' + webdate(now) );
    send( '</I></FONT></BODY></HTML>' );

    closeStdout;
    closeApp( application );

    end;
end;
end.
```

13. Browse Taxonomy

```
unit Browstx1; {Browse Taxonomy; created by Steve Earley.
        Last updated 20 Jun 96.}

interface

uses
 SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
 Forms, Dialogs, DB, Cgidb, Cgi, DBTables;

type
 TForm1 = class(TForm)
  CGIEnvData1: TCGIEnvData;
  DataSource1: TDataSource;
  Query1: TQuery;
  DataSource2: TDataSource;
  CGIDB1: TCGIDB;
  Query2: TQuery;
  procedure FormCreate(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form1: TForm1;

implementation

{$R *.DFM}


procedure TForm1.FormCreate(Sender: TObject);

var
  ProviderID : string;

begin
   with CGIEnvData1 do
   begin
      websiteINIFilename := paramstr(1);
```

```
application.onException := cgiErrorHandler;
application.processMessages;

ProviderID := GetSmallField('ProviderID');

{update Active Provider table}
with Query1 do
   begin
      close;
      SQL.clear;
      sql.add('UPDATE ACT_PROV ');
      sql.add('SET LastActionTime = ''' +DateTimeToStr(Now)+ '''');
      sql.add('WHERE ActProvID = ''' + ProviderID + '''');
      ExecSQL;
   end;

createStdout;
sendPrologue;

{standard header information}
send( '<HTML><HEAD>' );
sendTitle( 'DecisionNet Taxonomy' );
send( '</HEAD><BODY BGCOLOR="80B7B0">' );
send('<center><H1>DecisionNet Taxonomy</h1>');
send('This is the primary table used to generate the ');
send('categories for registering technologies.');
send('<p>');

{Open and display Taxonomy table}
with Query2 do
   begin
      close;
      SQL.Add('SELECT Parent,Child FROM TAXONOMY ');
      SQL.Add('ORDER BY Parent');
      open;
      CGIDB1.DrawTable;
      close;
   end;

send('</center>');
sendHR;
```

```
        {capture ProviderID, send user to Provider Menu}
        send('<CENTER>');
        send('<FORM method=post action="'+
            'http://131.120.39.63/cgi-win/dnet/provider/menu.exe">');
        send('<input type="hidden" name="ProviderID" '+
            'value="'+ProviderID+'"></td>');
        send('<input type="submit" value="Return to Provider Menu">');
        send('</form>');
        send('</CENTER>');

        sendHR;

        {Standard HTML footer information}
        send('<FONT SIZE=-1><I>');
        send( 'This application was created by Steve Earley');
        send( 'for Professor Hemant Bhargava.<br>' );
        send( 'Generated on ' + webdate(now) );
        send( '</I></FONT></BODY></HTML>' );

        closeStdout;
        closeApp( application );

    end;
end;
end.
```

D. SYSTEM ADMINISTRATOR SCRIPTS

1. Login Script

```
unit Login1; {Sysadmin Login; created by Steve Earley; updated: 5 Jun 96.}

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, Cgi, DBTables, DB;

type
  TForm1 = class(TForm)
    DataSource1: TDataSource;
    Table1: TTable;
    CGIEnvData1: TCGIEnvData;
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);

var
  SysadminID : string;
  sPassword : string;

begin
  with CGIEnvData1 do
  begin
    { required when this program runs under WebSite}
    webSiteINIFilename := paramstr(1);
```

```
application.onException := cgiErrorHandler;
application.processMessages;

{input fields from Login form (start.htm)}
SysadminID := getSmallField( 'SysadminID' );
sPassword := getSmallField( 'sPassword' );

{standard dynamic HTML header information}
createStdout;
sendPrologue;
send( '<HTML><HEAD>' );
sendTitle( 'DecisionNet SysAdmin Login' );
send( '</HEAD><BODY BGCOLOR="FFFFFF">' );

with Table1 do   {puts cursor on correct record in sysadmin table;
            GotoKey returns True if record is valid}
   begin
      open;
      SetKey;
      FieldByName('SysadminID').AsString := SysadminID;
      GotoKey;
   end;

if (Table1.FieldByName('sPassword').AsString <> sPassword) or
 (Table1.GoToKey = False) then   {password does not match sysadminID, or
                     user not in sysadmin table}
  begin
     send('<center><h1>Incorrect Login!</h1></center>');
     send('<h2>Please verify that your User Name and Password ' +
        'are correct, then try again.</h2>');
     send('<p>');
     sendHR;
     send('<p>');

     send('<CENTER> ' +
        '<TABLE BORDER=6 CELLPADDING=6> ' +
        '<TR ALIGN="CENTER" VALIGN=MIDDLE> ' +
              '<TD ALIGN="CENTER" VALIGN=MIDDLE> ' +
              '<A HREF="http://dnet.sm.nps.navy.mil/sysadmin/start.htm"> ' +
        'SysAdmin Start Page</A></TD>');
     send('</TR></TABLE></CENTER>');
  end
```

```
    else  {SysadminID matches sPassword}

      begin
         send('<center><h1>Welcome to DecisionNet!</h1>');
         send('<h2>You are logged in under the ' +
           'User Name <I>' + SysadminID  + '</I>.</h2></center>');
         send('<p>');
         sendHR;
         send('<p>');

         {capture SysadminID, send user to sysadmin Menu}
         send('<CENTER>');
         send('<FORM method=post action="'+
           'http://131.120.39.63/cgi-win/dnet/sysadmin/menu.exe">');
         send('<input type="hidden" name="SysadminID" '+
           'value="'+SysadminID+'"></td>');
         send('<input type="submit" value="SysAdmin Menu">');
         send('</form>');
         send('</CENTER>');
      end;

    Table1.close;

    {standard dynamic HTML footer information}
    send('<p>');
    sendHR;
    send('<p><i><font size=-1>');
    send( 'This application was created by Steve Earley ');
    send( 'for Professor Hemant Bhargava.<br>' );
    send( 'Generated on ' + webdate(now) );
    send( '</i></font></BODY></HTML>' );
    closeStdout;

  end;
end;
end.
```

2. System Administrator Menu

```
unit Menu1;    {SysAdmin menu; created by Steve Earley,
          Last updated 20 Jun 96}

interface

uses

 SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
 Forms, Dialogs, DB, Cgidb, Cgi, DBTables;

type
 TForm1 = class(TForm)
  CGIEnvData1: TCGIEnvData;
  procedure FormCreate(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);

var
  SysadminID: string;

begin
 with CGIEnvData1 do
  begin
    websiteINIFilename := paramstr(1);
    application.onException := cgiErrorHandler;
    application.processMessages;
    createStdout;
    sendPrologue;
```

```
{standard header information}
send( '<HTML><HEAD>' );
sendTitle( 'DecisionNet SysAdmin Menu' );
send( '</HEAD><BODY BGCOLOR="FFFFFF">' );

{Get ProviderID from start page; if user tries to go directly to
menu, raise an error}

SysadminID := GetSmallField('SysadminID');

if SysadminID = CGINotFound then
 begin
  send('<center><H1>Not Logged In</h1>');
  send('<h2>You are currently not logged in to DecisionNet. ');
  send('Please <a href="http://131.120.39.66/sysadmin/start.htm">login</a> ');
  send('to continue.</h2></center>');
 end

else
 begin
 send('<center><H1>DecisionNet SysAdmin Menu</h1></center>');
 sendHR;

{***MODIFY PASSWORD***}
 send('<H3>Change SysAdmin Password:</H3>');
 send('<CENTER> ');
 send('<TABLE BORDER=6 CELLPADDING=6>');
 send('<FORM METHOD = POST  ACTION= ');
 send('"http://131.120.39.63/cgi-win/dnet/sysadmin/modify.exe">');
send('<INPUT TYPE=HIDDEN NAME="SysadminID" VALUE="'+SysadminID+'">');

 send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
 send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
 send('Old Password: </TD>');
 send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
 send('<INPUT TYPE=PASSWORD NAME="sPassword" SIZE=20></TD>');

 send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
 send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
 send('New Password: </TD>');
 send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
 send('<INPUT TYPE=PASSWORD NAME="sPassword2" SIZE=20></TD>');
```

```
send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('Retype New Password : </TD>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<INPUT TYPE=PASSWORD NAME="sPassword3" SIZE=20></TD>');

send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<INPUT TYPE=SUBMIT VALUE="Change Password"></TD>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<INPUT TYPE=RESET VALUE="      Cancel      "></TD></FORM>');

send('</TR></TABLE></CENTER>');
sendHR;

{***VIEW TABLES***}
send('<H3>View Tables:</H3>');
send('<CENTER>');
send('<TABLE BORDER=6 CELLPADDING=6>');
send('<FORM METHOD=POST ACTION= ');
send('"http://131.120.39.63/cgi-win/dnet/sysadmin/viewtabl.exe">');
send('<INPUT TYPE=HIDDEN NAME="SysadminID" VALUE="'+SysadminID+'">');

send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<SELECT NAME="TableName" TYPE="text">');
 send('<OPTION VALUE="CONSUMER">Consumer');
 send('<OPTION VALUE="PROVIDER">Provider');
 send('<OPTION VALUE="TECHNOLO">Technology');
 send('<OPTION VALUE="ACT_CONS">Active Consumer');
 send('<OPTION VALUE="ACT_PROV">Active Provider');
 send('<OPTION VALUE="TECHINFO">Technology Info');
 send('<OPTION VALUE="ACT_NODE">Active Node');
 send('<OPTION VALUE="ACTXTECH">Active Excl Technology');
 send('<OPTION VALUE="TECHGRAP">Technology Graph');
 send('<OPTION VALUE="USEDTECH">Used Technology');
 send('<OPTION VALUE="CONSMIRR">Consumer Mirror');
 send('<OPTION VALUE="PROVMIRR">Provider Mirror');
 send('<OPTION VALUE="TECHMIRR">Technology Mirror');
 send('<OPTION VALUE="TAXONOMY">Taxonomy');

send('</SELECT></TD>');
```

```
send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<INPUT TYPE=SUBMIT VALUE="    View Table    " ALIGN="MIDDLE">');
send('</TD></FORM></TABLE>');
send('</CENTER>');
sendHR;

{***RUN SQL STATEMENT***}
send('<H3>Run SQL Statement:</H3>');
send('<OL><LI>Your command <I>must</I> start in the upper left-hand ');
send('space of the textarea below.');
send('<LI>Commands must follow the Borland Database Engine ');
send('<A HREF="http://dnet.sm.nps.navy.mil/sysadmin/syntax.htm" ');
send('TARGET="help">');
send('supported syntax</A>.');
send('<LI>If the BDE does not like your statement, you will receive a ');
send('<B>Capability not Supported</B> message. Just click on the Back ');
send('Arrow and try again.');
send('<LI>If you try to delete a parent record with referential ');
send('integrity constraints, you will not receive an error message, ');
send('and the record will remain intact.');
send('</OL>');

send('<CENTER>');
send('<FORM METHOD=POST ACTION= ');
send('"http://131.120.39.63/cgi-win/dnet/sysadmin/runsql.exe">');
send('<INPUT TYPE=HIDDEN NAME="SysadminID" VALUE="'+SysadminID+'">');
send('<textarea rows=10 cols=75 name="SQLCommand">');
send('</textarea>');
send('<P>');
send('<INPUT TYPE=SUBMIT VALUE="Run SQL Statement">');
send('<INPUT TYPE=RESET VALUE="Clear Fields"></FORM>');
send('</CENTER>');
sendHR;

{***RUN TIMEOUT SCRIPT MANUALLY***}
send('<H3>Run Timeout Script</H3>');
send('This is a manual (CGI) version of the hourly Timeout Program.');
send('<P><P>');
send('<FORM METHOD=POST ACTION= ');
send('"http://131.120.39.63/cgi-win/dnet/Sysadmin/timeout.exe">');
send('<CENTER>');
send('<INPUT TYPE=SUBMIT VALUE="Run Timeout Script" ALIGN=middle>');
```

```
    send('<INPUT TYPE=HIDDEN NAME="SysadminID" VALUE="'+SysadminID+'">');
    send('</FORM>');
    send('</CENTER>');
    end;
    sendHR;

    {standard footer information}
    send('<p><I><FONT SIZE=-1>');
    send( 'This application was created by Steve Earley ');
    send( 'for Professor Hemant Bhargava.<br>' );
    send( 'Generated on ' + webdate(now) );
    send( '</FONT></I></BODY></HTML>' );

    closeStdout;
    closeApp( application );

  end;
end;
end.
```

3. Change Password

```
unit Modify1; {SysAdmin Password Modification; created by Steve Earley;
        last update 7 Jun 96.}

interface

uses
 SysUtils, WinTypes, WinProcs, Messages, Classes,
 Forms, Cgi, DB, DBTables;

type
 TForm1 = class(TForm)
  DataSource1: TDataSource;
  Table1: TTable;
  CGIEnvData1: TCGIEnvData;
  DataSource2: TDataSource;
  Query1: TQuery;
  procedure FormCreate(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);

var
 SysadminID : string;
 sPassword : string;
 sPassword2 : string;
 sPassword3 : string;

begin
 with CGIEnvData1 do
 begin
```

```
{required when this program runs under WebSite}
webSiteINIFilename := paramstr(1);
application.onException := cgiErrorHandler;
application.processMessages;

{HTML page header info}
createStdout;
sendPrologue;

send( '<HTML><HEAD>' );
sendTitle( 'Modify SysAdmin Password' );
send( '</HEAD><BODY BGCOLOR="FFFFFF">' );

{receive input fields from HTML form generated by menu.exe}
SysadminID := getSmallField( 'SysadminID' );
sPassword := getSmallField( 'sPassword' );
sPassword2 := getSmallField( 'sPassword2' );
sPassword3 := getSmallField( 'sPassword3' );

if (sPassword2 = '') and (sPassword3 = '') then
  begin
    sPassword2 := sPassword;
    sPassword3 := sPassword;
   end;

with Table1 do    {puts cursor on correct record in Sysadmin table}
   begin
      open;
      SetKey;
      FieldByName('SysadminID').AsString := SysadminID;
      GotoKey;
   end;

if (Table1.FieldByName('sPassword').AsString <> sPassword) or
 (sPassword = cginotfound) then  {Password does not match SysadminID, or
                                  Password not entered}

  begin
    send('<center><h1>Incorrect Attempt!</h1>');
    send('<h2>Please verify that your Old Password ' +
      'is correct, then try again.</h2></center>');
    send('<p>');
    sendHR;
```

```
    send('<p>');

    {Capture SysadminID, send user back to Menu}
    send('<CENTER>');
    send('<FORM method=post action="'+
      'http://131.120.39.63/cgi-win/dnet/sysadmin/menu.exe">');
    send('<input type="hidden" name="SysadminID" '+
       'value="'+SysadminID+'"></td>');
    send('<input type="submit" value="Return to SysAdmin Menu">');
    send('</form>');
    send('</CENTER>');
  end

else  {SysadminID matches sPassword; ok to change information}
  begin
   if sPassword2 <> sPassword3 then {bad registration}
    begin
     send('<center><h1>Password Mismatch</h1>');
     send('<h2>Please verify your password choice, ');
     send('and try again.</h2></center>');
     send('<p>');
     sendHR;
     send('<p>');

     {Capture SysadminID, send user back to Menu}
     send('<CENTER>');
     send('<FORM method=post action="'+
      'http://131.120.39.63/cgi-win/dnet/sysadmin/menu.exe">');
     send('<input type="hidden" name="SysadminID" '+
      'value="'+SysadminID+'"></td>');
     send('<input type="submit" value="Return to SysAdmin Menu">');
     send('</form>');
     send('</CENTER>');
    end

   else  {correct sequence entered for changing password}
    begin
     sPassword := sPassword2;

     {Update Sysadmin table}
     with Query1 do
      begin
       close;
```

```
            SQL.clear;
            sql.add('UPDATE SYSADMIN ');
            sql.add('SET sPassword = ''' +sPassword+ '''');
            sql.add('WHERE SysadminID = ''' + SysadminID + '''');
            ExecSQL;
           end;

        {HTML page body for successful registration change}
        send('<center><h1>Sysadmin Password Accepted</h1>');
        send('<h2>Your information has been updated for ' +
            'User Name <I>' + SysadminID  + '</I>. ');
        send(' Thank you for using DecisionNet.</h2></center>');
        send('<p>');
        sendHR;
        send('<p>');

        {capture SysadminID, send user to Sysadmin Menu}
        send('<CENTER>');
        send('<FORM method=post action="'+
         'http://131.120.39.63/cgi-win/dnet/sysadmin/menu.exe">');
        send('<input type="hidden" name="SysadminID" '+
         'value="'+SysadminID+'"></td>');
        send('<input type="submit" value="SysAdmin Menu">');
        send('</form>');
        send('</CENTER>');
        send('<p>');

       end;
    end;

   {HTML page footer info}
   send('<HR><I><font size=-1>');
   send( 'This application was created by Steve Earley for Professor ');
   send( 'Hemant Bhargava.<br>' );
   send( 'Generated on ' + webdate(now) );
   send( '</font></i></BODY></HTML>' );
   closeStdout;

 end;
end;
end.
```

4. View DecisionNet Tables

```
unit Viewtab1;  {SysAdmin View Table Script; created by Steve Earley.
                 Last updated 12 Jul 96.}

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, DB, Cgidb, Cgi, DBTables;

type
  TForm1 = class(TForm)
    CGIEnvData1: TCGIEnvData;
    DataSource1: TDataSource;
    Query1: TQuery;
    CGIDB1: TCGIDB;
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);

var
  TableName : string;
  SysadminID : string;

begin
  with CGIEnvData1 do
    begin
      websiteINIFilename := paramstr(1);
      application.onException := cgiErrorHandler;
      application.processMessages;
```

```
createStdout;
sendPrologue;
TableName := GetSmallField('TableName');
SysadminID := GetSmallField('SysadminID');

{standard header information}
send( '<HTML><HEAD>' );
sendTitle( 'View DecisionNet Tables' );
send( '</HEAD><BODY BGCOLOR="FFFFFF">' );
send('<center><H1>View DecisionNet Tables</h1></center>');

{Display Table Name}
send('<H3>Table Name: ' + TableName + '</H3>');
send('<p>');

{Build table using simple query}
with query1 do
 begin
  close;
  SQL.clear;
  SQL.add('SELECT * FROM ' + TableName);
  open;
 end;

{Display table using CGIDB}
send('<center>');
CGIDB1.drawTable;
query1.close;
send('</center>');
sendHR;

{Give option to view another table}
send('<H3>View Another Table:</H3>');
send('<CENTER>');
send('<TABLE BORDER=6 CELLPADDING=6>');
send('<FORM METHOD=POST ACTION= ');
send('"http://131.120.39.63/cgi-win/dnet/sysadmin/viewtabl.exe">');
send('<INPUT TYPE=HIDDEN NAME="SysadminID" VALUE="'+SysadminID+'">');
send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<SELECT NAME="TableName" TYPE="text">');
 send('<OPTION VALUE="CONSUMER">Consumer');
 send('<OPTION VALUE="PROVIDER">Provider');
```

```
      send('<OPTION VALUE="TECHNOLO">Technology');
      send('<OPTION VALUE="ACT_CONS">Active Consumer');
      send('<OPTION VALUE="ACT_PROV">Active Provider');
      send('<OPTION VALUE="TECHINFO">Technology Info');
      send('<OPTION VALUE="ACT_NODE">Active Node');
      send('<OPTION VALUE="ACTXTECH">Active Excl Technology');
      send('<OPTION VALUE="TECHGRAP">Technology Graph');
      send('<OPTION VALUE="USEDTECH">Used Technology');
      send('<OPTION VALUE="CONSMIRR">Consumer Mirror');
      send('<OPTION VALUE="PROVMIRR">Provider Mirror');
      send('<OPTION VALUE="TECHMIRR">Technology Mirror');
      send('<OPTION VALUE="TAXONOMY">Taxonomy');
    send('</SELECT></TD>');
    send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
    send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
   send('<INPUT TYPE=SUBMIT VALUE="    View Table    " ALIGN="MIDDLE">');
    send('</TD></FORM></TABLE>');
    send('</CENTER>');
    sendHR;

    {capture UserID, send user to Sysadmin Menu}
    send('<CENTER>');
    send('<FORM method=post action="'+
       'http://131.120.39.63/cgi-win/dnet/sysadmin/menu.exe">');
    send('<input type="hidden" name="SysadminID" '+
       'value="'+SysadminID+'"></td>');
    send('<input type="submit" value="Return to SysAdmin Menu">');
    send('</form>');
    send('</CENTER>');

    {standard HTML footer information}
    send('<p>');
    sendHR;
    send('<p><FONT SIZE=-1><I>');
    send( 'This application was created by Steve Earley');
    send( 'for Professor Hemant Bhargava.<br>' );
    send( 'Generated on ' + webdate(now) );
    send( '</I></FONT></BODY></HTML>' );
    closeStdout;
    closeApp( application );
  end;
end;
end.
```

5. Run SQL Statement

```
unit Runsql1;  {SysAdmin Run SQL Script; created by Steve Earley.
               Last updated 8 Jun 96.}

interface

uses
 SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
 Forms, Dialogs, DB, Cgidb, Cgi, DBTables;

type
 TForm1 = class(TForm)
  CGIEnvData1: TCGIEnvData;
  DataSource1: TDataSource;
  Query1: TQuery;
  CGIDB1: TCGIDB;
  procedure FormCreate(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form1: TForm1;

implementation
{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);

var
  SysadminID : string;
  SQLMessage : TStringList;
  i: smallint;
  SelectFlag : string;

begin
  with CGIEnvData1 do
   begin
    websiteINIFilename := paramstr(1);
    application.onException := cgiErrorHandler;
```

```
application.processMessages;
createStdout;
sendPrologue;

{Receive input fields from SysAdmin menu}
SysadminID := GetSmallField('SysadminID');
SQLMessage := TStringList.create;
SQLMessage.clear;
getTextArea( 'SQLCommand', SQLMessage );

{standard header information}
send( '<HTML><HEAD>' );
sendTitle( 'Run SQL on DecisionNet' );
send( '</HEAD><BODY BGCOLOR="FFFFFF">' );
send('<center><H1>Run SQL on DecisionNet</h1></center>');

{Build query using string list}
with query1 do
 begin
  close;
  SQL.clear;
  for i := 0 to SQLMessage.count - 1 do
   SQL.add(SQLMessage.strings[i]);
 end;

{Check to see if first six characters are 'select'. If they are,
use Query1.open to display table, otherwise use Query1.ExecSQL and
display a message that statement was executed}
SelectFlag := system.copy(SQLMessage.strings[0], 0, 6);
if LowerCase(SelectFlag) = 'select' then
 begin
  Query1.open;
  send('<center>');
  CGIDB1.drawTable;
  Query1.close;
  send('</center>');
 end
else
 begin
  Query1.ExecSQL;
  send('<center><h2>Your SQL Statement has been accepted. ');
  send('You may check your results by viewing the table ');
  send('from the SysAdmin menu.</h2></center>');
```

```
    end;
    sendHR;

    {Give option to run another command}
    send('<H3>Run Another Command:</H3>');
    send('<CENTER>');
    send('<FORM METHOD=POST ACTION= ');
    send('"http://131.120.39.63/cgi-win/dnet/sysadmin/runsql.exe">');
    send('<INPUT TYPE=HIDDEN NAME="SysadminID" VALUE="'+SysadminID+'">');
    send('<textarea rows=10 cols=75 name="SQLCommand">');
    for i := 0 to SQLMessage.count - 1 do
      send(SQLMessage.strings[i]);
    send('</textarea>');
    send('<P>');
    send('<INPUT TYPE=SUBMIT VALUE="Run SQL Statement">');
    send('<INPUT TYPE=RESET VALUE="Clear Fields"></FORM>');
    send('</CENTER>');
    sendHR;

    {capture UserID, send user to Sysadmin Menu}
    send('<CENTER>');
    send('<FORM method=post action="'+
       'http://131.120.39.63/cgi-win/dnet/sysadmin/menu.exe">');
    send('<input type="hidden" name="SysadminID" '+
       'value="'+SysadminID+'"></td>');
    send('<input type="submit" value="Return to SysAdmin Menu">');
    send('</form>');
    send('</CENTER>');

    {standard HTML footer information}
    send('<p>');
    sendHR;
    send('<p><FONT SIZE=-1><I>');
    send( 'This application was created by Steve Earley');
    send( 'for Professor Hemant Bhargava.<br>' );
    send( 'Generated on ' + webdate(now) );
    send( '</I></FONT></BODY></HTML>' );
    closeStdout;
    closeApp( application );
  end;
end;
end.
```

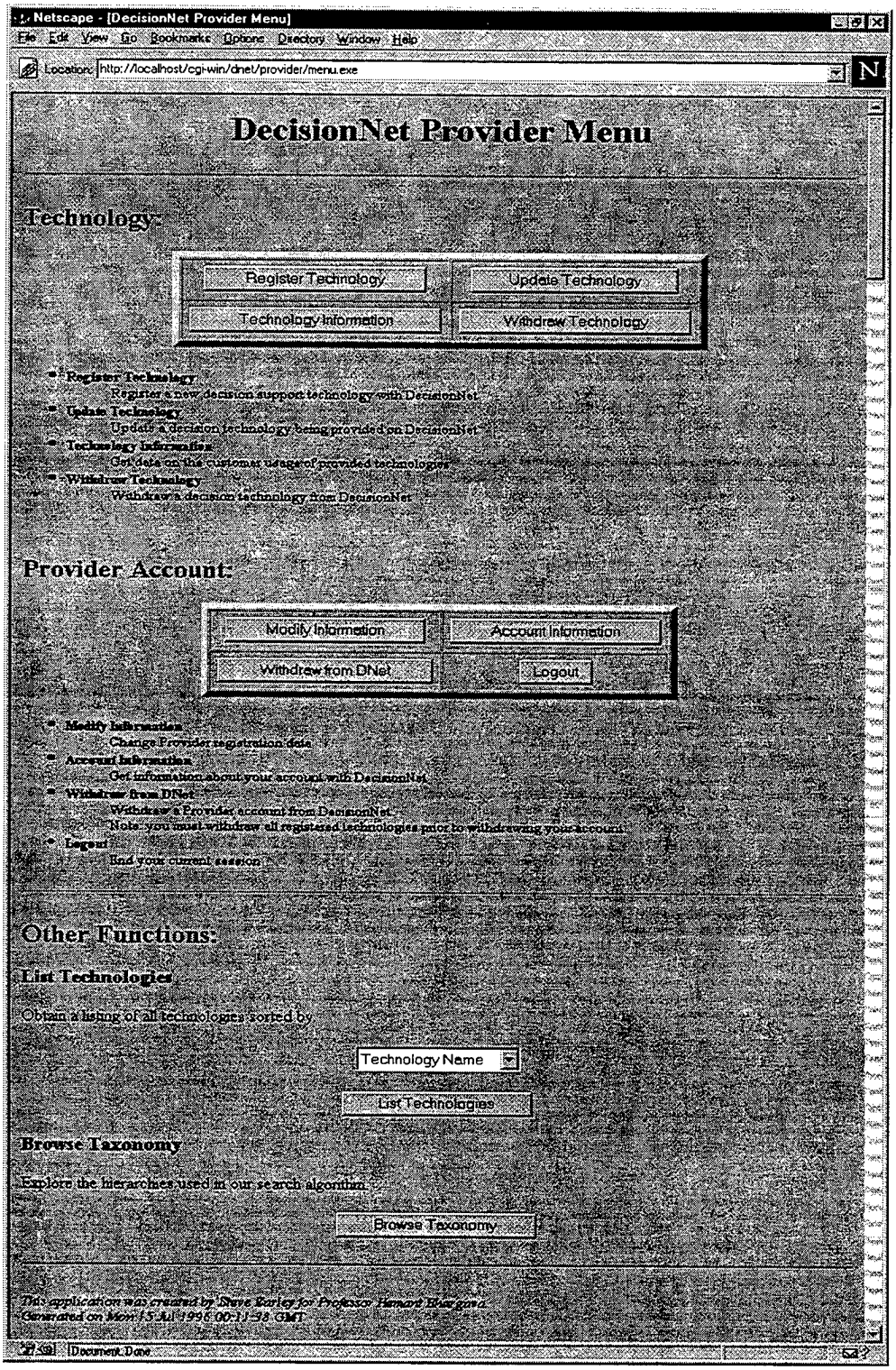
6. Timeout Script

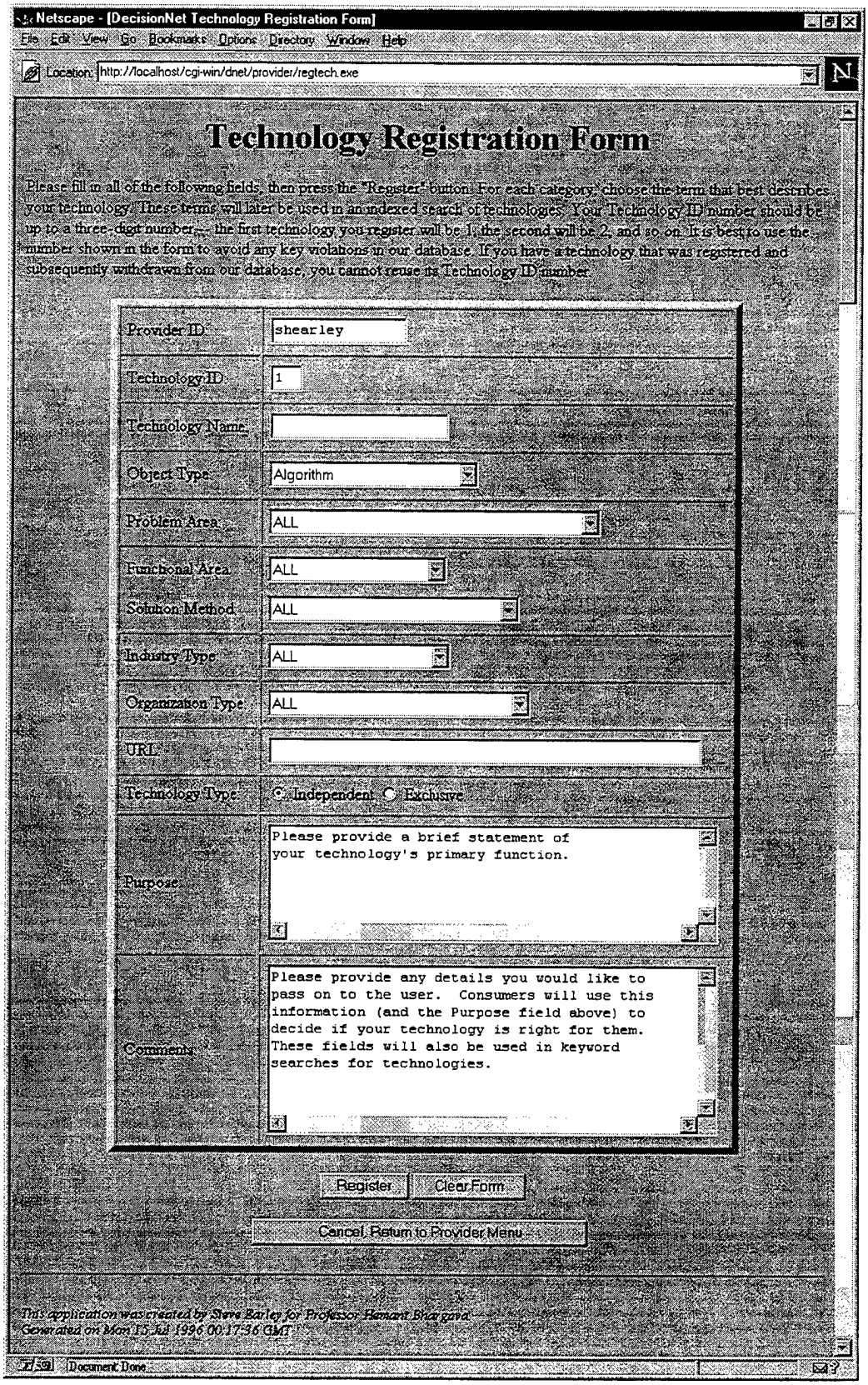
```
unit Timeout1; {Timeout script; created by Steve Earley; updated: 5 Jun 96.}

interface

uses
 SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
 Forms, Dialogs, Cgi, DBTables, DB;

type
 TForm1 = class(TForm)
  DataSource1: TDataSource;
  Table1: TTable;
  Table1ActConsID: TStringField;
  Table1SessionStartTime: TStringField;
  Table1LastActionTime: TStringField;
  DataSource2: TDataSource;
  Table2: TTable;
  CGIEnvData1: TCGIEnvData;
  procedure FormCreate(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form1: TForm1;

implementation
{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);

var
  SysadminID : string;
  ConsCount,ProvCount: Integer;

begin
 with CGIEnvData1 do
  begin
   {Required statements when program runs under WebSite}
```

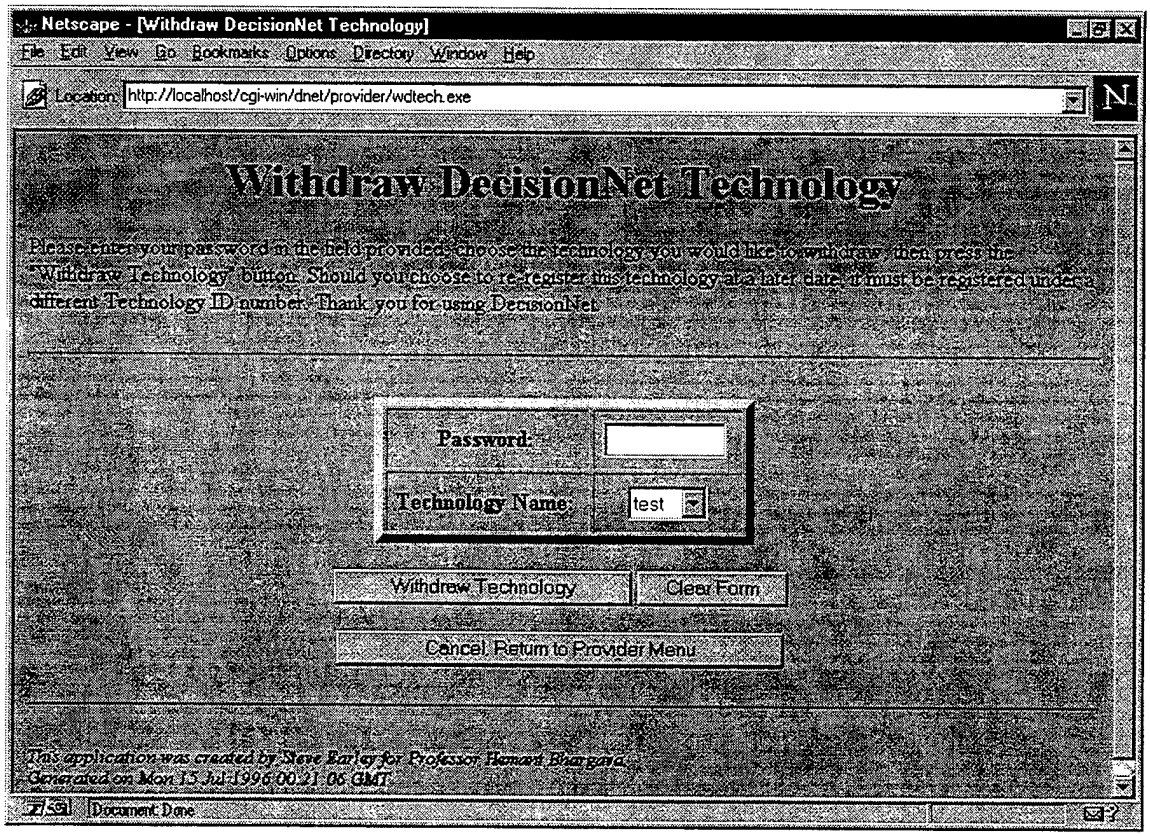
```
websiteINIFilename := paramstr(1);
application.onException := cgiErrorHandler;
application.processMessages;
createStdout;
sendPrologue;

{receive input field from Sysadmin Menu}
SysadminID := getSmallField( 'SysadminID' );

with Table1 do    {Check ACT_CONS for entries older than 6 hours}
 begin
  ConsCount := 0;
  open;
  first; {puts cursor on first record in table}

  while not EOF do
   begin
    if StrToDateTime(FieldByName('LastActionTime').AsString)
     < (Now - 0.25) then {current time minus 1/4 day}
      begin
       Delete; {record}
       ConsCount := ConsCount + 1;
      end
    else
      next; {puts cursor on next record}
   end;
  close;
 end;

with Table2 do    {Check ACT_PROV for entries older than 6 hours}
 begin
  ProvCount := 0;
  open;
  first; {puts cursor on first record in table}

  while not EOF do
   begin
    if StrToDateTime(FieldByName('LastActionTime').AsString)
     < (Now - 0.25) then {current time minus 1/4 day}
      begin
       Delete; {record}
       ProvCount := ProvCount + 1;
      end
```

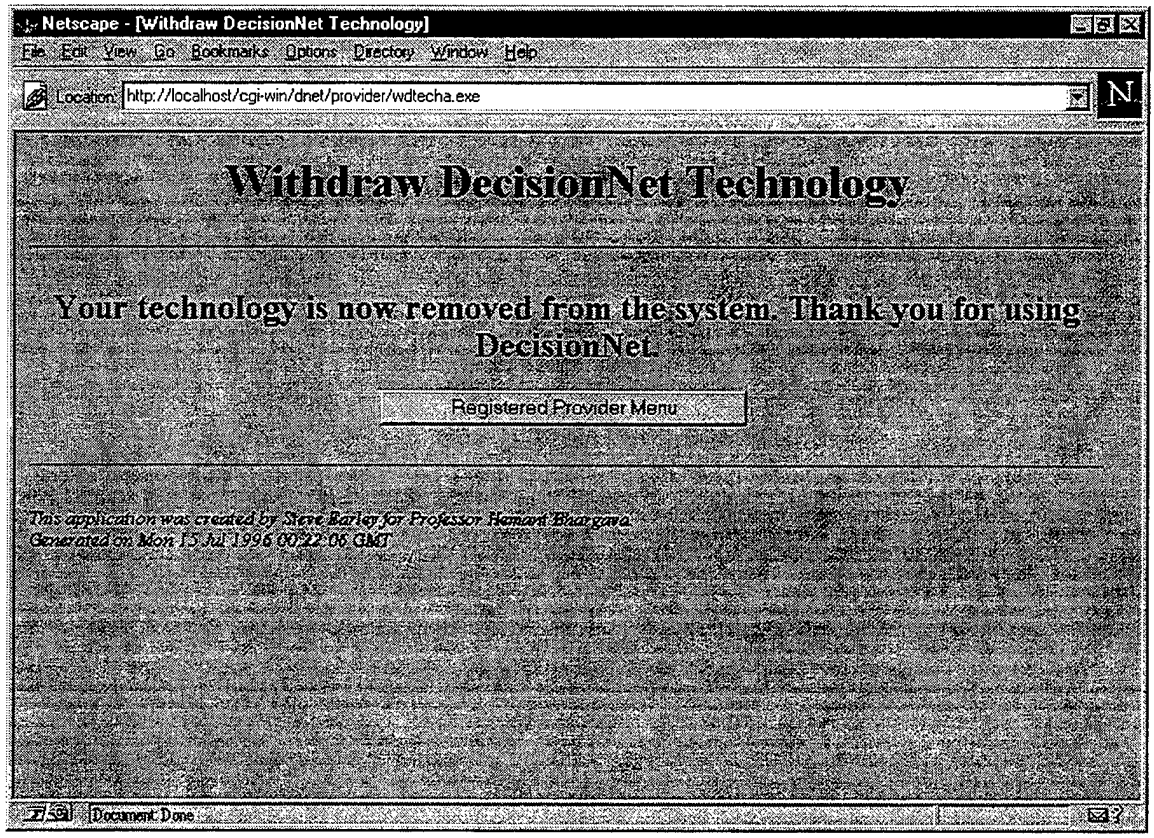
```
        else
          next; {puts cursor on next record}
        end;
      close;
    end;

  send( '<HTML><HEAD>' );
  sendTitle( 'DecisionNet Timeout Script' );
  send( '</HEAD><BODY BGCOLOR="FFFFFF">' );
  send('<center><H1>DecisionNet Timeout Script</h1>');
  send('Records have been successfully purged.<P>');
  send('ACT_CONS records deleted: ' + IntToStr(ConsCount) + '<P>');
  send('ACT_PROV records deleted: ' + IntToStr(ProvCount) + '<P>');
  send('<P>');
  sendHR;
  send('<P>');

  {Capture SysadminID, send user to Sysadmin Menu}
  send('<FORM method=post action="'+
   'http://131.120.39.63/cgi-win/dnet/Sysadmin/menu.exe">');
  send('<input type="hidden" name="SysadminID" '+
   'value="'+SysadminID+'"></td>');
  send('<input type="submit" value="Return to SysAdmin Menu">');
  send('</form>');
  send('</CENTER>');
  send('<p>');
  sendHR;

  send('<p><I><FONT SIZE=-1>');
  send( 'This application was created by Steve Earley ');
  send( 'for Professor Hemant Bhargava.<br>' );
  send( 'Generated on ' + webdate(now) );
  send( '</FONT></I></BODY></HTML>' );

  closeStdout;
  closeApp( application );

 end;
end;
end.
```

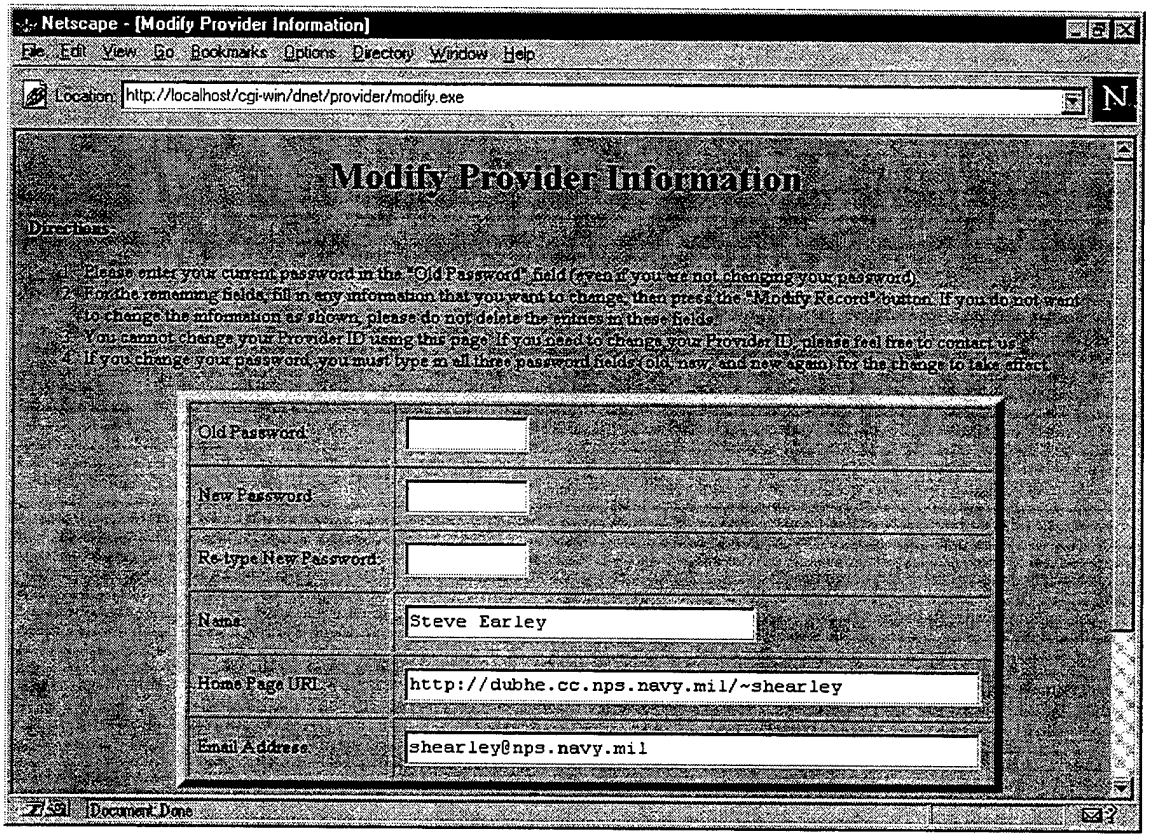
APPENDIX D. PROVIDER INTERFACE PAGES

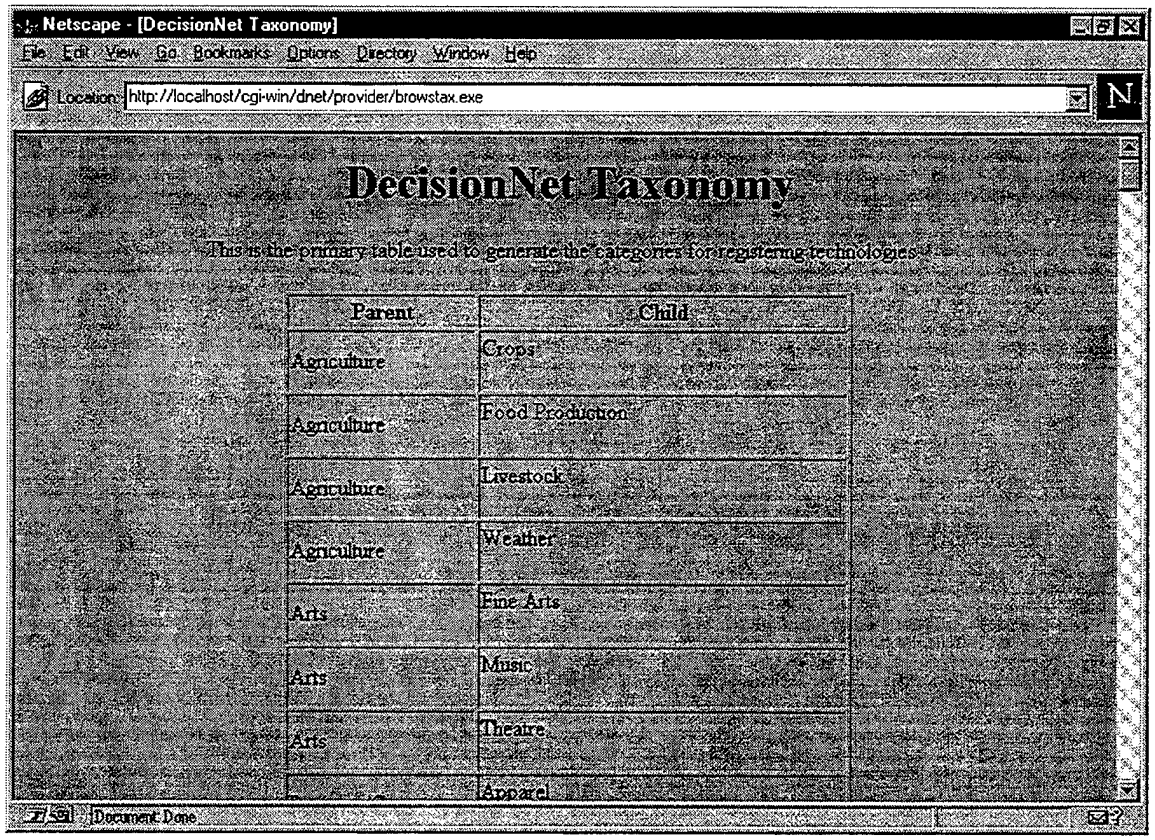



Netscape - [DecisionNet Technology Registration Form]
File Edit View Go Bookmarks Options Directory Window Help
Location: http://localhost/cgi-win/dnet/provider/regtech.exe
Technology Registration Form
Please fill in all of the following fields, then press the "Register" button. For each category, choose the term that best describes your technology. These terms will later be used in an indexed search of technologies. Your Technology ID number should be up to a three-digit number... the first technology you register will be 1, the second will be 2, and so on. It is best to use the number shown in the form to avoid any key violations in our database. If you have a technology that was registered and subsequently withdrawn from our database, you cannot reuse its Technology ID number.
Provider ID:
shearley
Technology ID:
1
Technology Name:
Object Type:
Algorithm
Problem Area:
ALL
Functional Area:
ALL
Solution Method:
ALL
Industry Type:
ALL
Organization Type:
ALL
URL:
Technology Type:
Independent
Exclusive
Purpose:
Please provide a brief statement of your technology's primary function.
Comments:
Please provide any details you would like to pass on to the user. Consumers will use this information (and the Purpose field above) to decide if your technology is right for them. These fields will also be used in keyword searches for technologies.
Register
Clear Form
Cancel, Return to Provider Menu
This application was created by Steve Barley for Professor Hemant Bhargava.
Generated on Mon 15 Jul 1996 00:17:56 GMT
Document Done

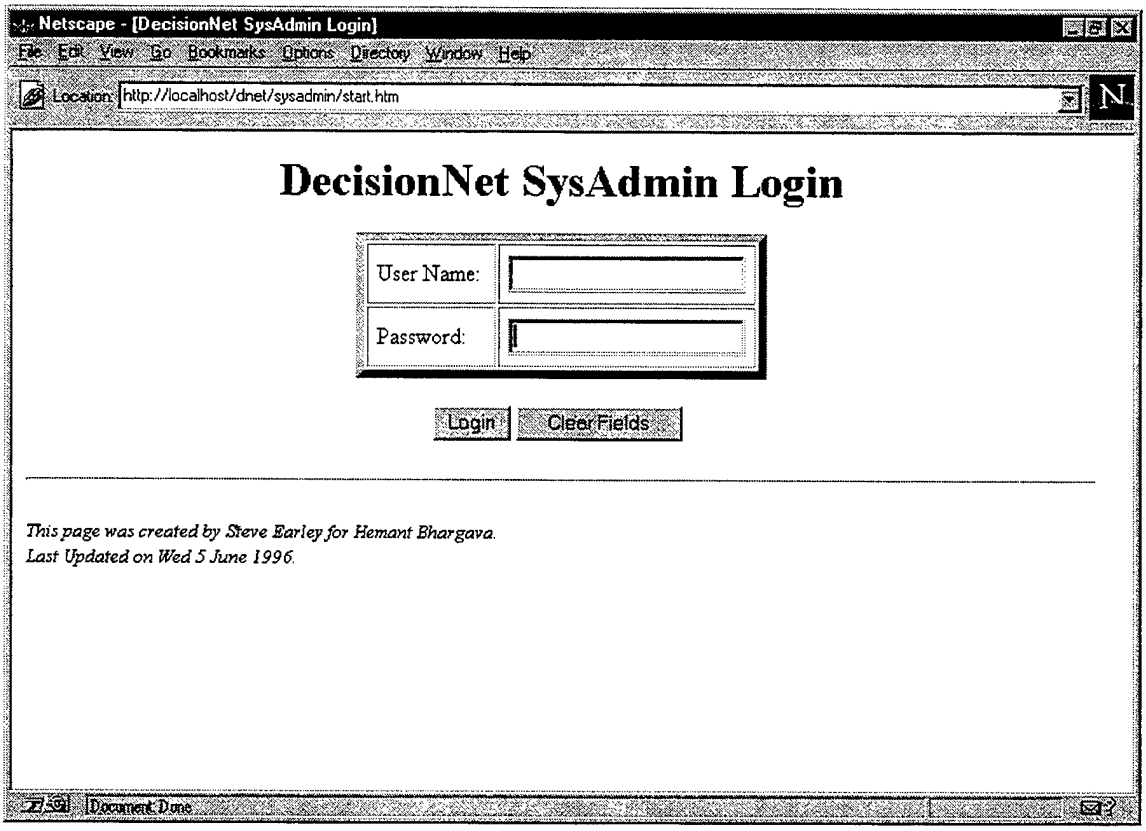

Netscape - [Withdraw DecisionNet Technology]
File Edit View Go Bookmarks Options Directory Window Help
Location: http://localhost/cgi-win/dnet/provider/wdtech.exe
Withdraw DecisionNet Technology
Please enter your password in the field provided, choose the technology you would like to withdraw, then press the "Withdraw Technology" button. Should you choose to re-register this technology at a later date it must be registered under a different Technology ID number. Thank you for using DecisionNet.
Password:
Technology Name:
test
Withdrew Technology
Clear Form
Cancel, Return to Provider Menu
This application was created by Steve Earley for Professor Hemant Bhargava
Generated on Mon 15 Jul 1996 00:21:06 GMT
Document Done

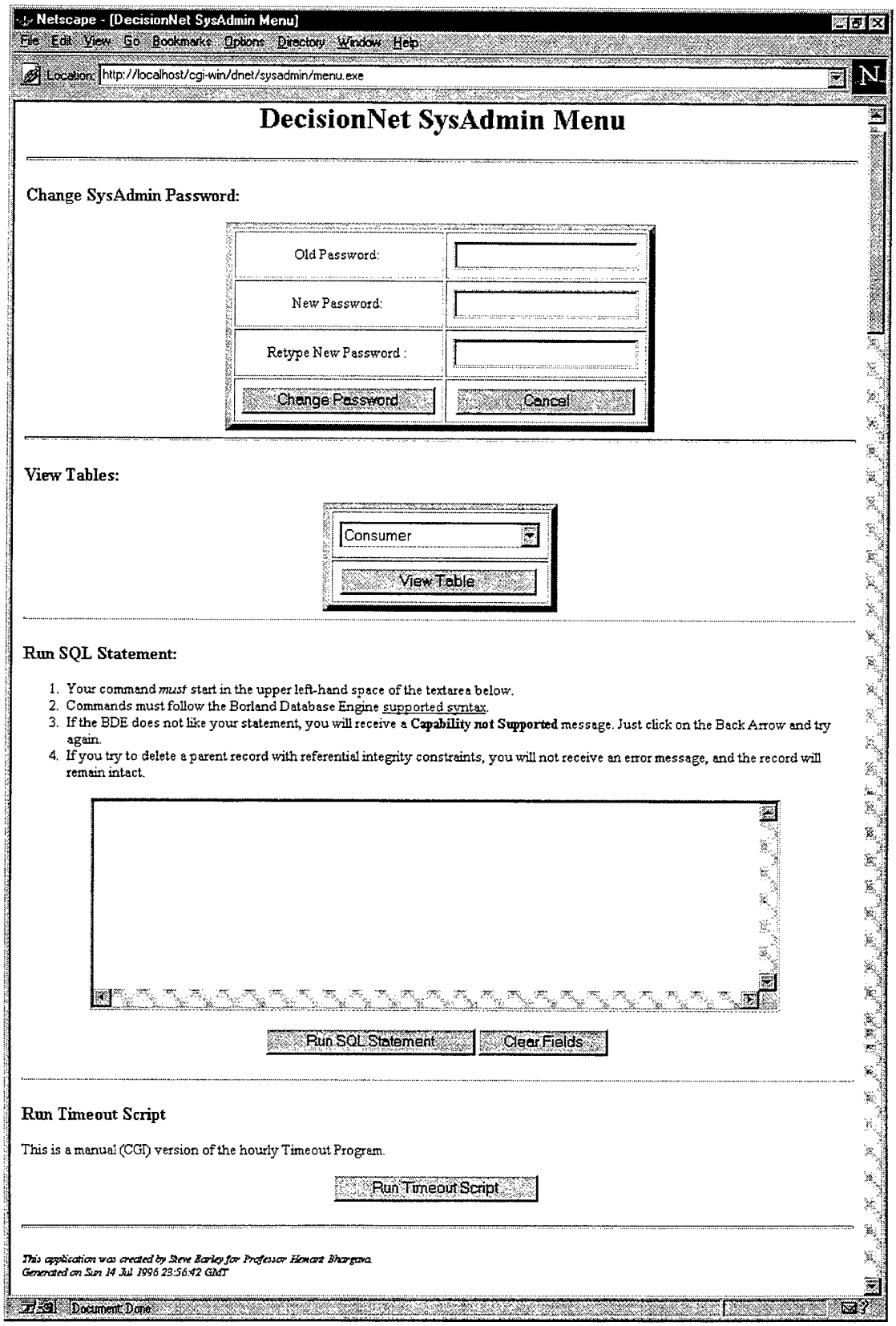

Netscape - [Withdraw DecisionNet Technology]
File Edit View Go Bookmarks Options Directory Window Help
Location: http://localhost/cgi-win/dnet/provider/wdtecha.exe
Withdraw DecisionNet Technology
Your technology is now removed from the system. Thank you for using DecisionNet.
Registered Provider Menu
This application was created by Steve Earley for Professor Hamant Bhargava
Generated on Mon 15 Jul 1996 00:22:06 GMT
Document Done

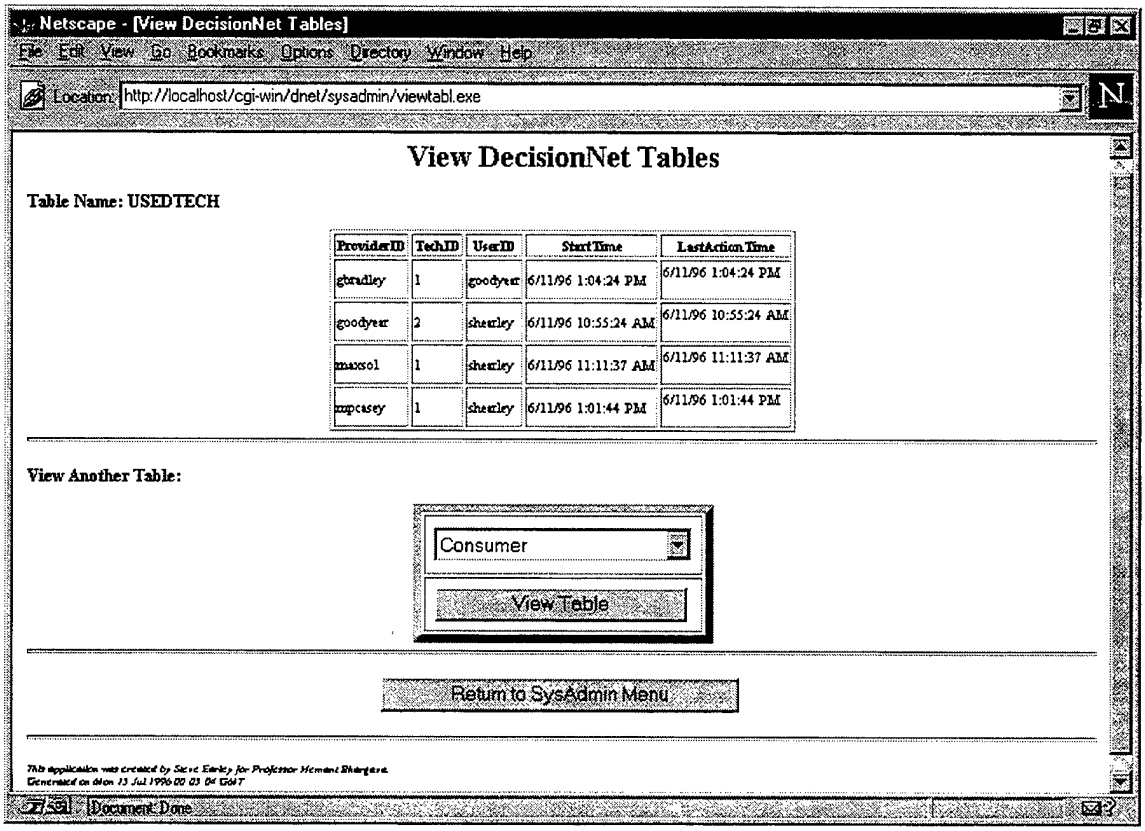

Netscape - [Modify Provider Information]
File Edit View Go Bookmarks Options Directory Window Help
Location: http://localhost/cgi-win/dnet/provider/modify.exe
Modify Provider Information
Directions:
1. Please enter your current password in the "Old Password" field (even if you are not changing your password).
2. For the remaining fields, fill in any information that you want to change, then press the "Modify Record" button. If you do not want to change the information as shown, please do not delete the entries in these fields.
3. You cannot change your Provider ID using this page. If you need to change your Provider ID, please feel free to contact us.
4. If you change your password, you must type in all three password fields (old, new, and new again) for the change to take effect.
Old Password:
New Password:
Re-type New Password:
Name:
Steve Earley
Home Page URL:
http://dubhe.cc.nps.navy.mil/~shearley
Email Address:
shearley@nps.navy.mil
Document Done

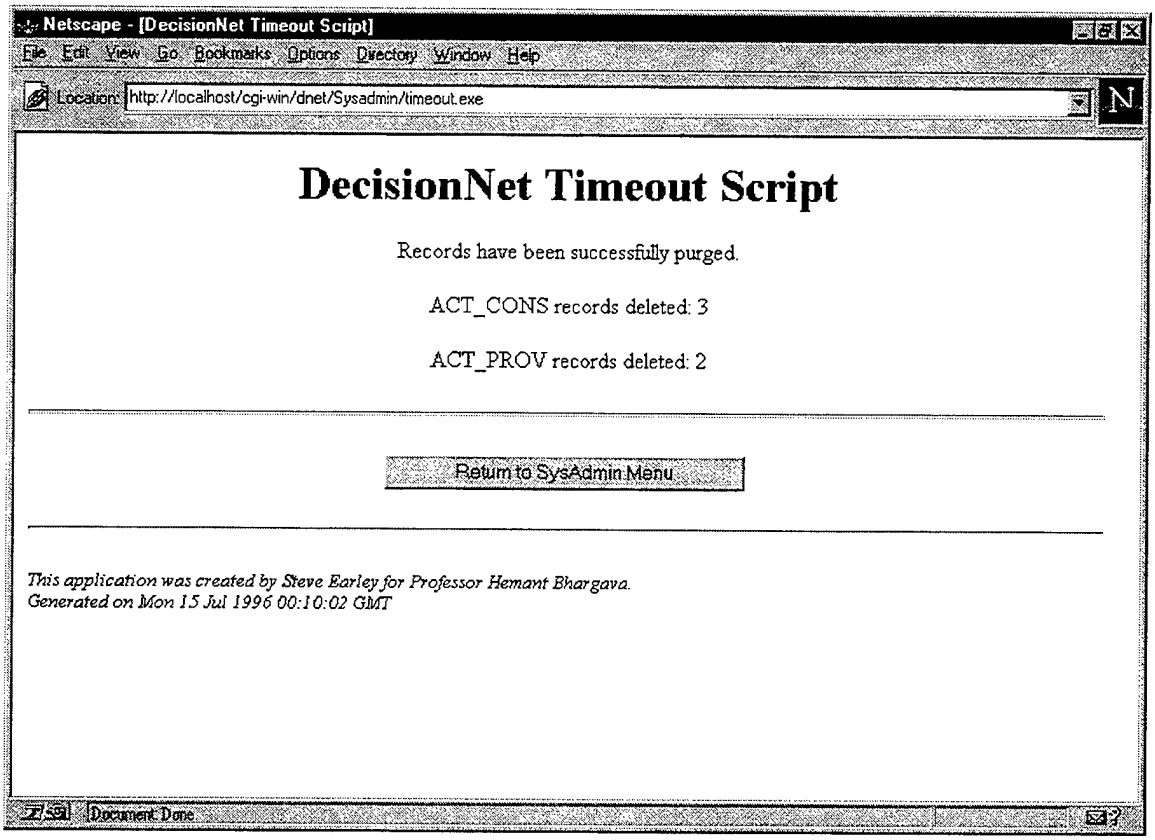

Netscape - [DecisionNet Taxonomy]
File Edit View Go Bookmarks Options Directory Window Help
Location: http://localhost/cgi-win/dnet/provider/browstax.exe
DecisionNet Taxonomy
This is the primary table used to generate the categories for registering technologies
Parent
Child
Agriculture
Crops
Agriculture
Food Production
Agriculture
Livestock
Agriculture
Weather
Arts
Fine Arts
Arts
Music
Arts
Theatre
Apparel
Document Done

APPENDIX E. SYSTEM ADMINISTRATOR INTERFACE PAGES




Netscape - [DecisionNet SysAdmin Menu]
File Edit View Go Bookmarks Options Directory Window Help
Location: http://localhost/cgi-win/dnet/sysadmin/menu.exe
DecisionNet SysAdmin Menu
Change SysAdmin Password:
Old Password:
New Password:
Retype New Password :
Change Password
Cancel
View Tables:
Consumer
View Table
Run SQL Statement:
1. Your command must start in the upper left-hand space of the textarea below.
2. Commands must follow the Borland Database Engine supported syntax.
3. If the BDE does not like your statement, you will receive a Capability not Supported message. Just click on the Back Arrow and try again.
4. If you try to delete a parent record with referential integrity constraints, you will not receive an error message, and the record will remain intact.
Run SQL Statement
Clear Fields
Run Timeout Script
This is a manual (CGI) version of the hourly Timeout Program.
Run Timeout Script
This application was created by Steve Barley for Professor Hemant Bhargava.
Generated on Sun 14 Jul 1996 23:56:42 GMT
Document Done


Netscape - [View DecisionNet Tables]
File Edit View Go Bookmarks Options Directory Window Help
Location: http://localhost/cgi-win/dnet/sysadmin/viewtabl.exe
View DecisionNet Tables
Table Name: USEDTECH
ProviderID TechID UserID StartTime LastActionTime
gbradley 1 goodyear 6/11/96 1:04:24 PM 6/11/96 1:04:24 PM
goodyear 2 sheurley 6/11/96 10:55:24 AM 6/11/96 10:55:24 AM
maxsol 1 sheurley 6/11/96 11:11:37 AM 6/11/96 11:11:37 AM
mpcasey 1 sheurley 6/11/96 1:01:44 PM 6/11/96 1:01:44 PM
View Another Table:
Consumer
View Table
Return to SysAdmin Menu
Document Done


Netscape - [DecisionNet Timeout Script]
File Edit View Go Bookmarks Options Directory Window Help
Location: http://localhost/cgi-win/dnet/Sysadmin/timeout.exe
DecisionNet Timeout Script
Records have been successfully purged.
ACT_CONS records deleted: 3
ACT_PROV records deleted: 2
Return to SysAdmin Menu
This application was created by Steve Earley for Professor Hemant Bhargava.
Generated on Mon 15 Jul 1996 00:10:02 GMT
Document Done

LIST OF REFERENCES

Bhargava, Hemant K., *About DecisionNet*, http://dnet.sm.nps.navy.mil/about.htm, May 1996.

Bhargava, Hemant K., Andrew S. King and Dan S. McQuay, "DecisionNet: An Architecture for Modeling and Decision Support Over the World Wide Web." *Proceedings of the Third International Conference on Decision Support Systems*, Vol. 2, June 1995.

Bhargava, Hemant K., Ramayya Krishnan, and Rudolf Müller, "On Parameterized Transaction Models for Agents in Electronic Markets for Decision Technologies." Article Submission to the Workshop on Information Technologies and Systems, Amsterdam, December 1995.

Bhargava, Hemant K. and Rudolf Müller, "Specification of DecisionNet Agents: Draft Notes from Monterey Discussion," working papers, Naval Postgraduate School, Monterey, CA, 13 October 1995.

Brownlee, David G., *Pricing Information Services in Electronic Markets: Case Study of DecisionNet*. Master's Thesis, Naval Postgraduate School, Monterey CA, March 1996.

Date, C. J., *An Introduction to Database Systems*, Vol. 1, 5th ed. New York: Addison-Wesley, 1990.

Delphi for Windows. Version 1. Computer Software. Borland International, Inc., Windows 3.1 or higher, disk, 1995.

King, Andrew S., *DecisionNet -- A Prototype Distributed Decision Support System Server*. Master's Thesis, Naval Postgraduate School, Monterey CA, September 1995.

Korth, Henry F., and Abraham Silberschatz, *Database System Concepts*, 2nd ed. New York: McGraw-Hill, 1991.

Lynnworth, Ann, "Delphi CGI Component". Computer Software. Windows 3.1 or higher, http://super.sonic.net/ann/delphi/cgicomp/detail.html, 1995.

Lynnworth, Ann, "Delphi CGIDB Component". Computer Software. Windows 3.1 or higher, http://super.sonic.net/ann/delphi/cgicomp/detail.html, 1995.

McQuay, Dan S., *A Framework for a Distributed Decision Support Network*. Master's Thesis, Naval Postgraduate School, Monterey CA, September 1995.

Palumbo, John R., *Financial Transaction Mechanisms for World Wide Web Applications*. Master's Thesis, Naval Postgraduate School, Monterey CA, March 1996.

Paradox for Windows. Version 5. Computer Software. Borland International, Inc., Windows 3.1 or higher, disk, 1994.

Rogers, Patricia M., *Indexing and Retrieval in Digital Libraries: Developing Taxonomies for a Repository of Decision Technologies.* Master's Thesis, Naval Postgraduate School, Monterey CA, March 1996.

Salsa for Windows. Student Edition, Version 1.1. Computer Software. Wall Data, Inc., Windows 3.1 or higher, disk, 1994.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center 8725 John J. Kingman Rd., Ste 0944 Ft. Belvoir, VA 22060-6218	2
2.	Dudley Knox Library Naval Postgraduate School 411 Dyer Rd. Monterey, CA 93943-5101	2
3.	Professor Hemant Bhargava (Code SM/BH) Naval Postgraduate School Systems Management Department Monterey, CA 93940-5000	3
4.	Professor Suresh Sridhar (Code SM/SR) Naval Postgraduate School Systems Management Department Monterey, CA 93940-5000	1
5.	Professor Gordon Bradley (Code OR/BZ) Naval Postgraduate School Operations Research Department Monterey, CA 93940-5000	1
6.	Professor Ted Lewis (Code CS/LT) Naval Postgraduate School Computer Science Department Monterey, CA 93940-5000	1
7.	Professor Gary Porter (Code CC/PO) Naval Postgraduate School Joint C4I Academic Group Monterey, CA 93940-5000	1
8.	Professor Dan Boger (Code SM/BO) Naval Postgraduate School Systems Management Department Monterey, CA 93940-5000	1

9. CAPT George Zolla, USN (Ret.) 1
Naval Postgraduate School
Monterey, CA 93943-5000

10. US Army Artificial Intelligence Center 1
OSIDC4, Attn SAIS-AI
107 Army Pentagon - Room 1D659
Washington, DC 20310-0107

11. Professor Ramayya Krishnan 1
The Heinz School
Carnegie Mellon University
Pittsburgh, PA 15213

12. Professor Rudolf Müller 1
Institut für Wirtschaftsinformatik
Humboldt-Universität zu Berlin
Spandauer Str. 1
10178 Berlin, Germany

13. Professor A. M. Geoffrion 1
Graduate School of Management
University of California
405 Hilgard Avenue
Los Angeles, CA 90024-1481

14. Professor Ramesh Sharda 1
College of Business Administration
Oklahoma State University
Stillwater, OK 74078

15. Mr. Clark Pritchett 1
U. S. Coast Guard Research and Development Center
1082 Shennecossette Avenue
Groton, CT 06340

16. CAPT Dan McQuay, USMC 1
1 Westwood Court
Stafford, VA 22135

17. Mrs. Susan K. Maslyk 1
5708 Longford Drive
Dublin, OH 43017

18. LT Steven H. Earley
Class 146
Surface Warfare Officers School Command
446 Cushing Road
Newport, RI 02841-1209

2